Winter 2014

Network Interface Design for Network-on-Chip

Jiawei Zhong
University of New Hampshire, Durham

NETWORK INTERFACE DESIGN FOR NETWORK ON CHIP

By

Jiawei Zhong

B.S., Nanjing University of Posts & Telecommunications, China, 2012

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

in

Electrical Engineering

December, 2014

This thesis has been examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering by:

Thesis Director, Qiaoyan Yu, PhD.
Assistant Professor of Electrical and Computer Engineering

Kent A. Chamberlin, PhD.
Professor of Electrical and Computer Engineering

Andrzej Rucinski, PhD.
Professor of Electrical and Computer Engineering

Radim Bartos, PhD.
Associate Professor of Computer Science

On December 12, 2014

Original approval signatures are on file with the University of New Hampshire Graduate School.

Acknowledgement

I would like to express my sincere gratitude to many people for their generous help during my graduate study at University of New Hampshire.

First, I would like to thank my advisor, Dr. Qiaoyan Yu, for her insightful guidance and technical training through my two years' research and study. I also thank her for providing me this unique opportunity to learn state-of-art VLSI design.

I want to thank all my committee members for their kind support during my thesis: thanks to Dr. Kent Chamberlin for modifying my proposal and giving me valuable advice on my research topic; thanks to Dr. Andrzej Rucinski for his guidance and help when I was doing teaching assistant; thanks to Dr. Radim Bartos for helping me to present my research work and prepare my thesis.

I am also very grateful to other professors and researchers at UNH, especially Dr. Lacourse, Dr. Messner, Dr. Kun, Dr. Smith, Dr. Miller and Dr. Sharif. They helped me a lot in both coursework and academic. I also want to thank all my colleagues and friends: Patrick Nsengiyumva, Ranita Bera, Neeraj Gill, Herman Pretorious and Drew Stock, for their help and support in coursework and research.

I wish to thank the ECE department at UNH for providing me financial assistantship to support my study and research. Finally, I would like to thank my parents for their unconditional love and support.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ASIC	Application Specific Integrated Circuit
CAD	Computer-aided design
CMOS	Complementary metal–oxide–semiconductor
FPGA	Field Programmable Gate Array
HT	Hardware Trojan
IC	Integrated Circuit
IP	Intellectual Property
NI	Network Interface
NoC	Network-on-Chip
SoC	System-on-Chip
VLSI	Very Large Scale Integration

Abstract

NETWORK INTERFACE DESIGN FOR NETWORK ON CHIP

by

Jiawei Zhong

University of New Hampshire, December, 2014

In the culture of globalized integrated circuit (IC, a.k.a chip) production, the use of Intellectual Property (IP) cores, computer aided design tools (CAD) and testing services from un-trusted vendors are prevalent to reduce the time to market. Unfortunately, the globalized business model potentially creates opportunities for hardware tampering and modification from adversary, and this tampering is known as hardware Trojan (HT). Network-on-chip (NoC) has emerged as an efficient on-chip communication infrastructure. In this work, the security aspects of NoC network interface (NI), one of the most critical components in NoC is investigated and presented. Particularly, the NI design, hardware attack models and countermeasures for NI in a NoC system are explored.

An Open Core Protocol compatible NI is implemented in an IBM0.18μm CMOS technology. The synthesis results are presented and compared with existing literature. Second, comprehensive hardware attack models targeted for NI are presented from system level to circuit level. The impact of hardware Trojans on NoC functionality and performance are evaluated. Finally, a countermeasure method is proposed to address the hardware attacks in NIs.

CHAPTER 1

INTRODUCTION

1.1 Background of On-Chip Communication Infrastructure

Current rapid improvement of VLSI technology is yielding a more powerful, capable and flexible system on a single silicon die. The embedded and computing system design nowadays has moved from single core to the era of multi-core and eventually to many core architectures [1]. The integration of numerous intellectual property (IP) blocks on a chip has become a feasible and popular design methodology, which is known as system-on-chip (SoC). One SoC example is shown in Figure 1-1.These heterogeneous IPs can be micro-processor, data memory, multimedia decoder and general peripherals. They mainly communicate with each other via an on-chip bus. Several industrial bus standards are available for SoCs, such as ARM AXI [2], IBM CoreConnect [3] and Wishbone [4].

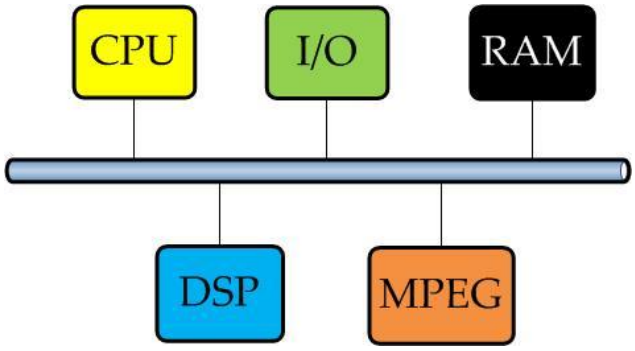


Figure 1-1 SoC: Bus-centric communication architecture

The traditional bus centric communication in multi-core SoC (MPSoC) has several drawbacks, including limited bandwidth efficiency, high latency, poor

flexibility and extensive overhead in terms of power and area [5]. Those limitations become more severe as the chip complexity and the number of IP cores increase. An emerging on-chip interconnect solution, known as network-on-chip (NoC), has been widely investigated by industry and academia communities [6-9]. The design of NoC and its interface to SoC are pivotal in addressing all the issues mentioned above. As a result, NoC is recognized as the mainstream communication architecture for MPSoCs. In NoC-based MPSoCs, multiple processing IP cores are connected within a network of routers and network interfaces (NIs), rather than the regular buses. Figure 1-2 shows a generic NoC system typically consists of routers, links and network interfaces (NIs). Routers are the places where the data is directed to different paths, links are wires connecting two routers and the NIs are the units implementing the protocol and sockets between routers and IPs [6].

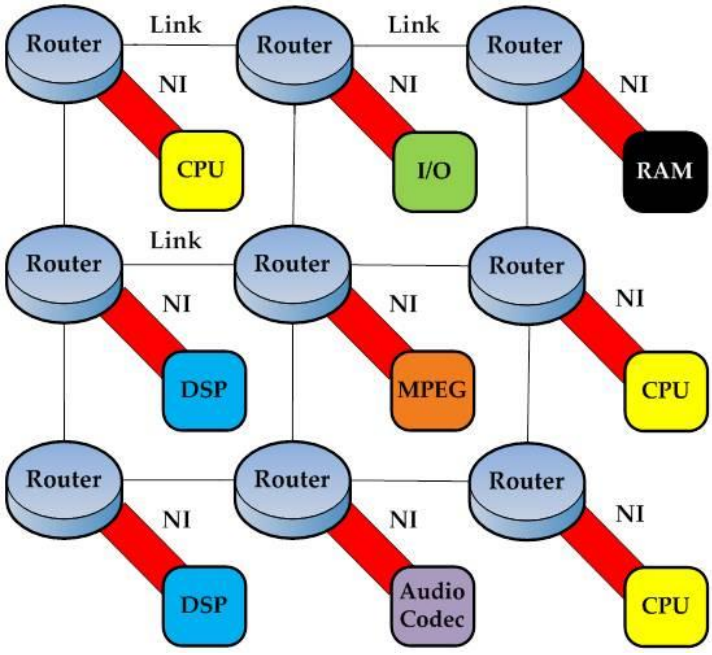


Figure 1-2 Topological illustration of Network-on-Chip (NoC)

NoC has superior advantages such as scalability, reusability and high

performance over traditional buses. However, this new infrastructure also brings in new weaknesses to the MPSoC system [7]. In most of the previous literature, the research on NoC mainly focuses on system topology, functionality implementation, routing & switching techniques, traffic characterization, computer-aided design (CAD) tools and libraries design [10-13], but security and reliability aspects in such system have not been fully explored. Therefore, in this thesis, we address the hardware security issues in NoC design, and particularly in the network interface.

1.2 Motivations for Trustworthy IC

Hardware security in IC is emerging as an important research topic in recent years. Modern IC design and manufacturing often involve purchased IP cores from third-party vendors, electronic design automation (EDA) software from different suppliers, outsourced design, testing, assembling and packaging services. This globalization business model helps IC companies to reduce cost and shorten the time to market, but also makes fabricated ICs vulnerable to several malicious attacks [14-15]. Figure 1-3 shows the trust level at different steps of an IC design cycle. Each party associated with the design cycle can potentially compromise an IC's functionality by intentional modification and tempering, which is known as hardware Trojan (HT).

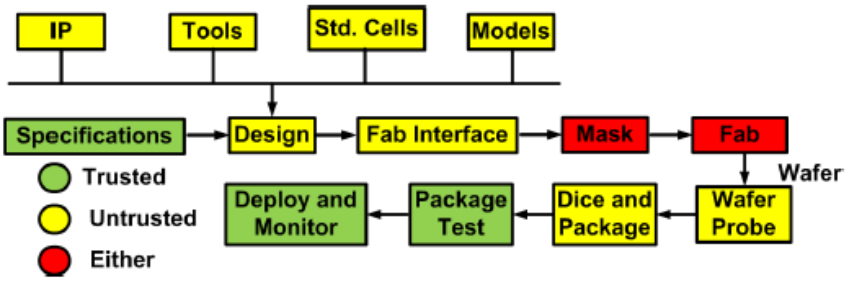


Figure 1-3 Trust level at each level of an IC design cycle (Source: [8])

HTs can be introduced in any design phase of IC design, from RTL description, gate-level netlist, CAD tool libraries to GDSII file [16-17]. HTs can have serious impact on the IC while in operation, especially in security sensitive applications such as military, communication and national infrastructure. For instances, a malicious microprocessor with shadow mode was implemented in [9]; a back door insertion into the MicroSemi chips was reported in [10], which brought severe economic loss; and similar attacks are also found in military radar recently [11], which made the radar fail to detect enemies. As more and more transistors can be integrated into a single die, we may expect that hardware security aspect in communication infrastructure, such as the prevalent NoC, will be another concern in the near future. The consequences of HT attacks in NoC, particularly in NI, may be more severe than those in IP cores for the following reasons:

- NoC is a complex network composed of a large number of heterogeneous IP cores. In modern IC business, some of these IP cores may be possibly purchased from third-party vendors. It is difficult to ensure that all of the IPs is trustworthy, thus increasing the risk of hardware Trojan insertion [21-22].
- Although the IP cores on-chip are heterogeneous, the NoC infrastructure itself is

highly modular. The routers and NIs in a NoC can be identical in a network [23-24]. If hardware Trojan is inserted into the NoC standard module or library, the HT attacks can be easily manifested through the entire network.

- The NIs in NoC are not only connecting wires between IP cores and routers. The functionalities of state-of-art NIs include packet encapsulation and de-capsulation, routing computation, quality-of-service (QoS) and flow control [12]. As the NIs becomes more powerful and complicated, more attack locations and opportunities are available to adversary, and the consequences of HTs are more difficult to predict than before as well.

Most of the previous works related to hardware trustworthy and security are mainly focused on the functional IP cores, such as general-purpose processor [9], cryptographic IP, memory block and peripheral interface [62]. Although simple examples for the attacks in NoCs have been provided in previous studies [11, 26-29], those examples are not sufficient and complete for NoC designers to create meaningful attack models at physical level to evaluate the security performance for a given NoC design.

1.3 Related Work &Our Contributions

Security aspects in NoCs were first discussed in [11,26]. Security attacks on integrated circuits and their application systems are generally classified into software attacks, physical attacks and side-channel attacks. Such classification also applies to NoCs. Security attacks in NoCs cause denial of service, extract secret information, or

alter execution or configuration of a system to conduct additional duties for adversaries (a.k.a hijacking) [13].

Diguet et al. proposed a first solution for configurable NoC based communication system in [14]. Diguet's security framework is composed of secure network interface and secure configuration manager, not including NoC routers. To implement a secure NoC, Fiorin et al. [15] highlighted the need of address protection unit in network interface and weighted round robin arbiter in router, as well as security automata to monitor unexpected routines. Recently, Fiorin et al. presented data protection unit designs in details to address the secure memory access in NoC [30-31]. To tackle power, electromagnetic and network snooping attacks on NoCs, Gebotys and Zhang made an effort on the transport and application layers by securing the cryptographic key exchange mechanism in NoC [16-17]. Sajeesh and Kapoor also exploited authenticated encryption in network interface to secure communication among IP cores [18]. LeMay and Gunter introduced a NoC Firewall implemented in a special functional hardware description language to facilitate formal analysis and thus detected attacks that violate NoC protocol [19]. As cryptography units are hardware-consuming, not all NoC-based systems can afford using crypto units at transport and application layers of NoCs. To broaden the choice of methods for NoC security enhancement, security attacks on other NoC layers need to be investigated, as well. Some tangible examples of security attacks are provided in existing literatures [13, 20]; however, the attack models for NoCs are either not thorough or too

high-level to be directly used in the procedure of NoC security assessment. Specific HT attack models that are tightly couple with typical NoC design details will benefit NoC designers and users to be aware of potential HT attacks and the corresponding HT consequence in system development stage.

Besides, NI design and implementation are extensively explored in [23,25,34-39], but none of them ever addresses the hardware Trojan or security aspects in NI. In this work, we will fill in the gap. Our main contributions are as follows.

- We design and implement a highly modular network interface to facilitate the standard OCP compatible NoC design. Hardware cost, latency performance and power consumption of our baseline NI are compared to existing NIs.
- We analyze the NI from system level to gate level and propose potential HT attack models in terms of possible attack locations and potential low probability trigger signals. The attack models are justified either by analytical derivation or by practical simulation.
- The impact of HT in a NoC system is quantitatively investigated and visually evaluated by applying the attack models to NoC applications.
- Finally, we propose HT aware detection method for HT countermeasures embedded into the baseline design. The efficiency of our method is also assessed with quantitative results.

1.4 Thesis Organization

The rest of the thesis is organized as below.

Chapter 2 briefly describes the basics of NoC, including architecture, routing scheme, switching technique, end-to-end flow control and advanced functionalities.

Chapter 3 introduces the proposed NI design in details, including all the sub-modules.

Chapter 4 illustrates a comprehensive hardware Trojan attack model for NI, including HT trigger design and payload placement.

Chapter 5 presents implementation details of several practical HTs. The impact of HTs on video and image applications is also evaluated using application. A countermeasure method is also proposed to facilitate the HT detection.

Chapter 6 summarizes this thesis and discusses future work.

CHAPTER 2

NOC BASICS

NoC encompasses a wide spectrum of research topics, ranging from highly software application related issues (e.g. network traffic characterization and routing scheme), across system topology (e.g. NoC topologies) to physical link level implementation (e.g. FPGA and ASIC).The design space of a NoC is considerably larger than that of a bus-based solution, as NoC can employ different routing and arbitration strategies can be implemented as well as different organizations of the communication infrastructure. This enables the SoC designer to find one of suitable NoC solutions for specific system characteristic and constraints. In this chapter, we present the main concepts involved in NoCs. Our baseline NI design was implemented in IBM 0.18μm technology following the digital ASIC design flow. The synthesis results for silicon area, power and timing are compared with existing literature.

2.1 NoC Structure

The conceptual NoC is shown in Figure 2-1.A generic Network-on-Chip system typically consists of network interfaces (NI), routers, physical links, and intellectual property (IP) cores [21]. The IP cores are also called functional blocks or processing elements, which are the main on-chip resources for data computation and processing.

IP core can be CPU, DSP processor, on chi memory, general I/O block and video codec.

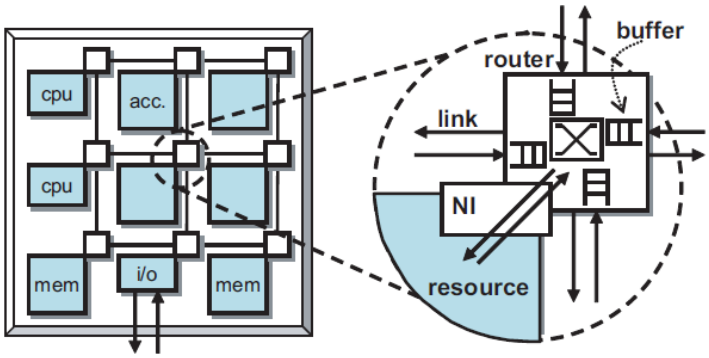


Figure 2-1 NoC: Network-based communication architecture (Source: [22])

2.1.1 Router

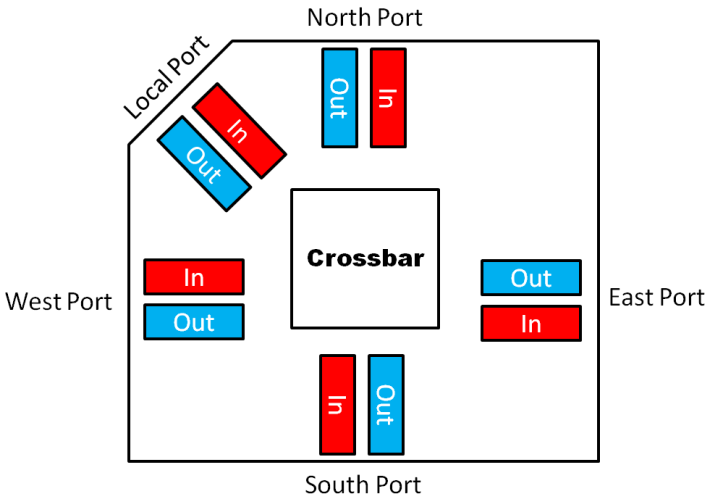


Figure 2-2 Five-port NoC router architecture

The design of router is to route the data according to chosen protocols, routing and switching techniques. A NoC router is composed of input ports and output ports connected to the shared NoC channels, a switching crossbar connecting the input ports to the corresponding output ports, and a local port to access the IP core at this router node. A generic architecture of five-port NoC router can be seen in Figure 2-2 [41-42]. In addition to this physical connection infrastructure, the router also contains

a logic block that implements the flow control policies and defines the overall strategy for transferring data through the NoC.

2.1.2 Links

Links provide the raw bandwidth and physical connection with routers. It is composed of one or more logical or physical channels, and each channel consists of a set of metal wires. Typically, a NoC link has two physical channels, making a full-duplex connection for bi-directional transmission between the routers. The number of parallel wires per channel is typically uniform throughout the network and is known as the channel bandwidth.

2.1.3 Network Interface (NI)

Network Interfaces (NIs) implement the logic connection by which IP cores connect to the NoC. Their function is to decouple the data processing of IP core from the communication network, and make it feasible to reuse core and communication infrastructure. In [23], the author partitions the NI into two parts: a front end and a back end. The front end handles the core requests and is ideally unaware of the NoC. This part is usually implemented as interconnect socket and some of the industrial communication protocols are applied at the front end, such as OCP [24], AXI [2], and Wishbone [4]. The back end part assembles and disassembles the packet, reorders buffers, implements synchronization protocols, and helps the router in terms of storage. A generic NI architecture is shown as Figure 2-3.

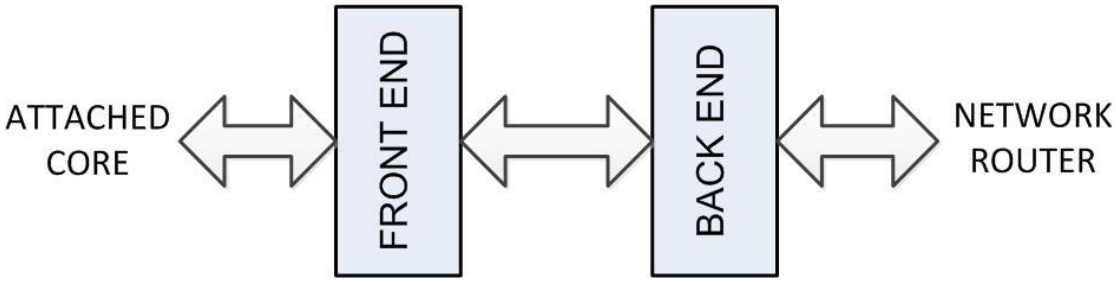


Figure 2-3 Generic NI architecture

2.2 NoC Design Problems

To provide a complete NoC solution, we need specify many aspects and characteristics from bottom to up. The designers may also need to trade off among functionality, overhead and reliability according to the application requirements. The state-of-art NoCs adapt mangy advanced networking features and functionalities [25]. The main design issues of NoCs include packet format, NoC topology, switching technique, routing strategies, flow control and quality of service (QoS) requirement [12].

2.2.1 Packet Format

Packet format defines the structure of the basic data units in a NoC system. Since the data flow in NoC is based on packet switching, the format will make a significant difference to the hardware design. Typically, a packet will be further divided into several sequential flits, which are the minimum data units in NoC communication [23]. A packet will consist of header flit, payload flit and tail flit. The header flit usually contains key information of a packet, such as source and destination; the payload flit only carries the general transmitted data and tail flit is used to

differentiate the boundary of two packets. The flit format is illustrated in Figure 2-4.

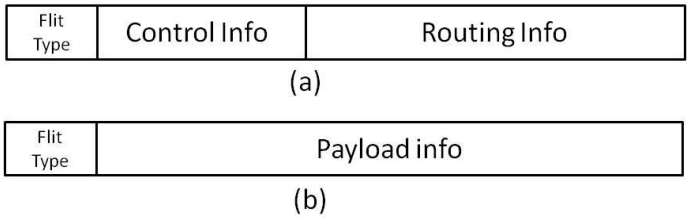


Figure 2-4 Generic Packet Flit Format (a) Header Flit Format (b) Payload & Tail Flit Format

2.2.2 Topology

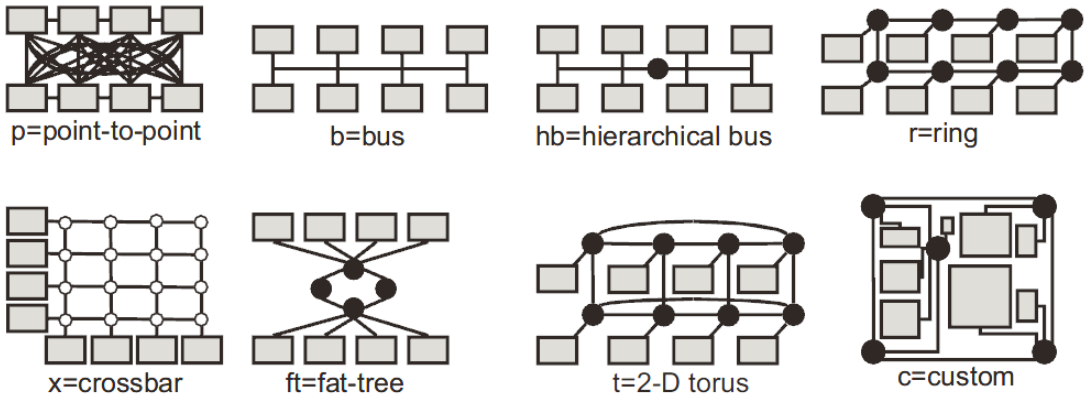


Figure 2-5 Basic NoC Topologies (Source: [22])

Topology defines how NoCs are organized. Each node is composed of a router, a NI and an IP core. From the communication perspective, there are various topologies for NoC architectures. These topologies include mesh, torus, ring, octagon, tree and other irregular interconnection networks. Some sample topologies are shown in Figure 2-5. 2-D mesh topology is adopted in most NoC designs, as it is proved to be more efficient in terms of latency, power consumption and ease of implementation than other topologies [22].

2.2.3 Packet Switching Techniques

Packet switching technique determines how the packets are forwarded and transferred through the NoC network. When the first NoC was proposed in early

2000s, circuit switching was preferred. To execute circuit switching, a path from source to destination is informed prior to transfer by reserving the routers and links before packet transfer. All data will follow the exact same path and the path is torn down after the packet transfer is completed. Currently, most of NoC implementations are based on packet switching. In packet switching, all the packets are transmitted without reserving the entire path. Packet switched networks can further be classified as wormhole, store and forward, and cut through [25].

2.2.4 Routing Scheme

Routing schemes are different from the concept of switching techniques. A routing algorithm defines the exact route or path for the data from source to destination rather than the way how the data are transferred. Routing scheme can be generally classified into deterministic routing and adaptive routing. In the deterministic routing, all packets follow the same route between a given pair of source and destination and data deadlock can be easily avoided. Unlike deterministic routing, adaptive routing may dynamically update or change the route for packets to reduce congestion, but requires special attention to avoid deadlock and livelock. In current NoC design, most packet switched networks utilize deterministic routing such as XY routing [6].

2.2.5 Flow Control

End-to-end flow control characterizes the packet movement along the NoC. A NoC system may have lots of buffer or memory locally inside the NIs and routers.

When the packets are transferring through the network, they are stored temporarily in the buffering memory and wait for next step processing. In this sense, flow control is necessary to guarantee that these buffers will not overflow. Traditional end to end flow control schemes have credit based scheme and ACK based scheme [25].

2.2.6 Quality-of-Service

Quality-of-Service (QoS) refers to the levels of guarantees given for data transfer. Goossensens et al. define two types of QoS in [26]: best-effort (BE) and guaranteed service (GS). With best-effort NoCs, only completion of the communication is ensured and data are transferred as soon as they are ready; with guaranteed service NoCs, some extra services or properties are ensured, such as the correctness of data, the completion of transaction and the error free communication.

2.2.7 On-chip Interconnect Protocols

As mentioned in 2.1, the front end of network interface typically is implemented as industrial on-chip standards or protocols. This feature allows NI to separate the IP cores from the whole network. As a result, designers can focus on the design of various processing IPs without concerning the integration of NoC with IP cores. The IP cores can be in a “plug-and-play” fashion for better reusability and performance [27].

However, the on-chip communication protocols were first designed for SoC applications and bus centric connection. Integrating heterogeneous IP cores in a SoC often requires the insertion of glue logic, and standards of on-chip bus protocols were

developed to avoid this problem [28]. As we are migrating to the era of multi-processor SoC (MPSoC) and NoC, the on-chip protocols are also being modified and developed to adapt the demands of NoC. This section overviews the popular standardized on-chip interconnect protocols such as AMBA AXI [2], CoreConnect, Wishbone [4] and Open Core Protocol (OCP) [3].

- *AMBA (Advanced Micro-controller Bus Architecture)* is a standard devised by ARM Limited to support efficient on-chip communication for ARM processors. AMBA is hierarchically organized into system and peripheral bus segments, mutually connected via bridges. AMBA does not define the method of arbitration, instead, it allows the arbiter to be designed to meet the applications needs. AMBA is also a multi-bus system and several distinct buses are defined in the AMBA specification: ACE (Advanced Coherency Extensions), AHB (Advanced High-performance Bus), APB (Advanced Peripheral Bus) and AXI (Advanced eXtensible Interface)[2]. Nowadays, AMBA AXI is already one of the leading on-chip interconnect systems used in high-performance FPGAs, MPSoCs and NoCs.
- *CoreConnect*is a bus architecture developed by IBM to ease the integration and reuse of processors, system and peripheral cores [3]. CoreConnect is hierarchically comprised of a processor local bus (PLB), an on-chip peripheral bus (OPB) and a device control register bus (DCRB). These three buses provide an efficient interconnection of cores, library macros, and custom logic within a SoC

or MPSoC.

- *Wishbone* bus architecture was first developed by Silicore Corpation, and now is maintained by OpenCores organization [4]. In 2003, it was released to public for free. Wishbone does not define hierarchical buses, but defines the master and slave interfaces. It also supports different types of transaction, such as read/write. Designer can choose the arbitration mechanism to fit the application need.
- *Open Core Protocol* is a standard defined by OCP-IP. Unlike other bus centric architecture, OCP is a comprehensive, bus-independent and configurable interface standard between IP cores and on-chip communication subsystems. It is also openly licensed and a designer can select only those signals and features from the palette of OCP configurations to fill the need.

CHAPTER 3

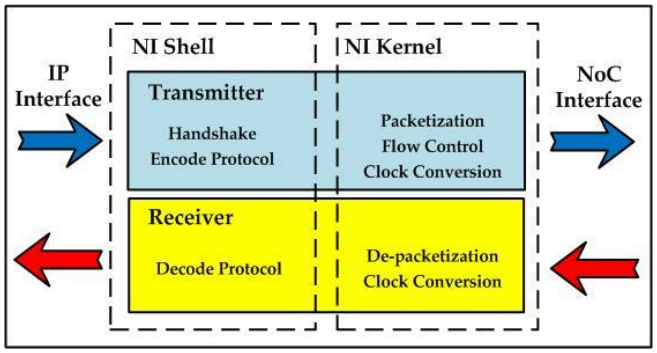
PROPOSED NI DESIGN

The NI design guidelines were summarized and a generic NI architecture was presented in [23], and various NI implementations have been extensively reported in [23,25, 34-35, 37-39].The state-of-art NI design includes more than the basic functionalities of synchronization and protocol wrapping. Advanced networking functionalities, such as routing schemes, quality-of-service (QoS), flow control and error management, begin to be incorporated into the NI. Different hardware optimizations for latency, power and area are also considered in the previous works[12]. In this chapter we illustrate the details of our proposed NI baseline design based on previous works.

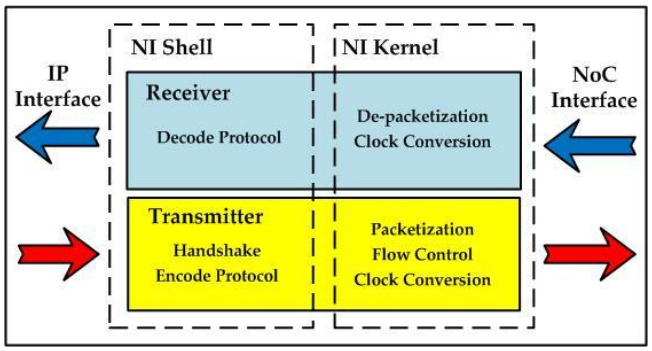
3.1 Overview of Generic NI Design

IP cores are typically categorized into Master and Slave IPs [23]. In a NoC infrastructure, Master IPs are active processing elements such as Reduced Instruction Set Computing (RISC) or DSP processors. Slave IPs are passive data recipients such as memory and I/O peripherals. Since NoC is known as a transaction-based and message-passing communication paradigm, only Master IPs can generate transaction requests and Slave IPs only receive the requests and send back proper responses to Master IPs. To meet different demands of Master and Slave, bi-directional Initiator and Target NIs are designed for Master and Slave IPs, respectively. Initiator NIs are

connected to Master IPs to convert IP request transactions into NoC traffic and translate the packets received from network; in contrast, Target NIs only receive requests from network and respond the Master according to the requests.



(a)



(b)

Figure 3-1 Overview of network interface (a) Initiator NI (b) Target NI

As shown in Figure 3-1, Initiator and Target NIs present a mirrored architecture to each other. There are two fundamental separations in a generic NI structure: each NI contains an upstream transmitter path and a downstream receiver path; kernel and shell are also distinguished for NoC and IP core specified issues, respectively. A NI shell implements the socket protocol to handshake with IP cores and forward the requests and data to next-stage kernel. Therefore, a protocol specific shell is needed

for each IP core connected to the NoC. A NI kernel packetizes data from shell, and at the same time, depacketizes the response from the receiving path. A state-of-art NI kernel also handles flow control and routing. The separation of kernel and shell minimizes the effort of supporting various sockets when needed and keeping the kernel structure unchanged. In other words, whichever protocol, bus size, clock frequency that the IP core is using, all modules in the infrastructure may communicate with each other [12]. The transmitter and receiver paths are loosely coupled: whenever a request is processed, the receive path is notified; whenever a response is received, the transmit path is unblocked.

3.2 Application of OCP to NI Design

Table 3-1 Comparison of On-chip Protocols

Standard	CoreConnect	AMBA	OCP	Wishbone
Interconnect	Shared Bus	Shared Bus	Point-to-Point	Crossbar/Shared Bus/Point-to-Point
Bus Width	32-256	32-256	Configurable	8-64
Device	FPGA/PLD/ ASIC	FPGA/PLD/ ASIC	FPGA/PLD/ ASIC	PLD/ASIC
Application	High Speed Embedded System	High Speed Embedded System	High Speed/Low Cost Embedded System	High Speed/Low Cost Embedded System
License	Authorization Needed	Authorization Needed	Free	Free

As IP blocks are characterized typically by memory-mapped interfaces, most NoCs are based on message-passing networks. This demands core-centric sockets and the corresponding protocol wrapper between the shell and IP cores. In 2.2.7, we overview the most popular on-chip protocols including AMBA, CoreConnect,

Wishbone and OCP. Table 3-1 provides a more detailed comparison of their features.

Among all these standards, we adopt OCP in our particular design for several reasons. First, it is freely distributed and no license is needed; second, OCP is bus independent and core centric, and this feature is more suitable in separating the core from network fabric in a NoC design than other standards; third, OCP is highly flexible and configurable, and allows designer to select appropriate signals and features according to designs. OCP is increasingly popular in the NoC design and has potential to be utilized in future NoC benchmarks [29].

3.2.1 OCP Signaling and Encoding

The OCP protocol defines a point-to-point interface between Master and Slave IP cores. Only the master can presents request commands and the slave responds to the commands either by accepting data or providing feedback. OCP facilitates basic transfer mode with optional extensions such as burst, tag and thread. Burst extension has been proved to be a key feature of current NI design given its high bandwidth usage and low jitter [30]. A data packet is further partitioned into multiple small data flits in a burst transaction, so the transmission does not need to wait until enough space is available for an entire packet. Therefore, the latency is reduced and the speed increases.

Our design supports the precise burst over imprecise burst to facilitate further hardware optimization. The burst length is known before at the start of the burst, and it can be either single burst or multiple data. The detailed signaling of OCP is

presented in Table 3-2. Basic OCP signals include MCmd, MData, MAddr, SCmdAccept, SResp and SData. The designer can also select different extension signals according to demands.

Table 3-2 OCP Specification Signals

Group	Name	Width	Driver	Function
Basic	MAddr	configurable	master	Transfer Address
	MCmd	3	master	Transfer Command
	MData	configurable	master	Transfer Data
	MDataValid	1	master	Write Data Valid
	SDataAccept	1	slave	Accept Write Data
	SCmdAccept	1	slave	Accept Command
	SData	configurable	slave	Transfer Data
	SResp	2	slave	Transfer Response
Burst	MBurstLength	configurable	master	Transfer Burst Length
	MBurstSeq	3	master	Transfer Burst Sequence
	MReqLast	1	master	Last Write Request
	MDataLast	1	master	Last Write Data
	SRespLast	1	slave	Last Read Response
Threads	MConnID	configurable	master	Connection Identifier
	MThreadID	configurable	master	Request Thread Identifier
	SThreadID	configurable	slave	Response Thread Identifier
	MDataThreadID	configurable	master	Write Data Thread Identifier
Sideband	SInterrupt	1	slave	Slave Interrupt

3.2.2 OCP Timing Diagram

OCP specification also defines the timing diagram for various data transfers such as simple read and write, non-post write, read with handshake and response [24]. In our particular design, we mainly use burst based read and write transaction.

As shown in Figure 3-2, the Master IP starts the burst write by driving WR on MCmd, the first address of the burst on MAddr, valid data on MData, the burst length

on MBurstLength. The Slave IP will assert SCmdAccept as soon as the MData is accepted.

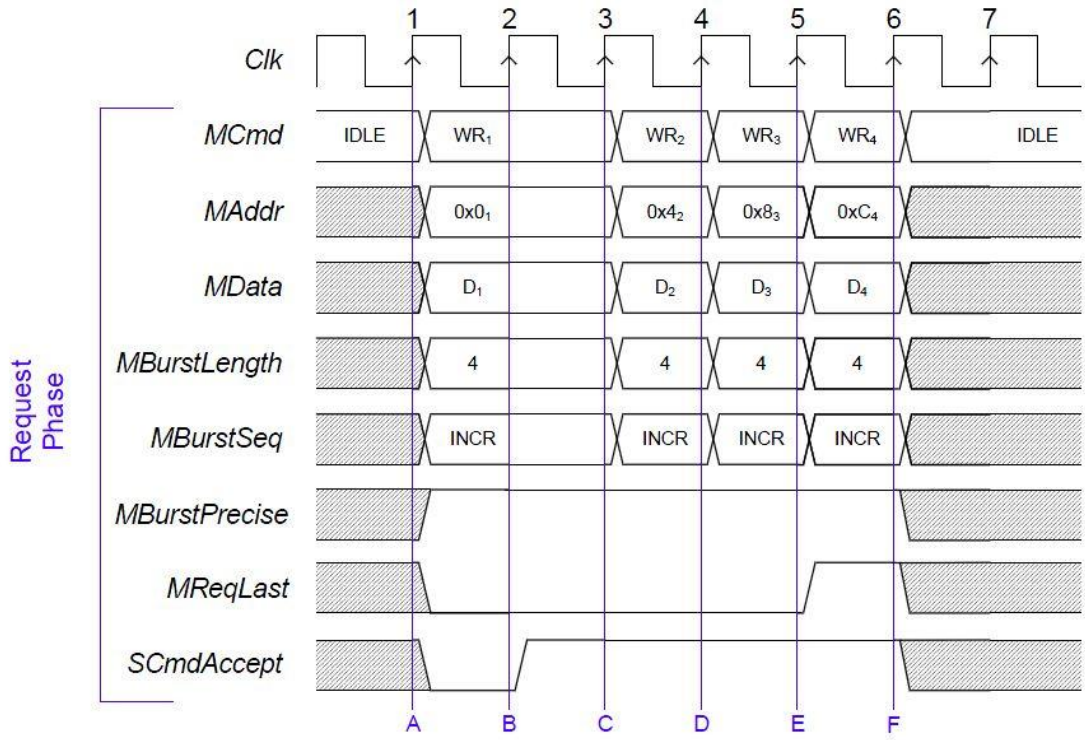


Figure 3-2 Timing diagram for burst write transaction (Source: [24])

In the burst read transaction, the Master IP starts a read request by driving RD on MCmd, a valid address on MAddr, MBurstLength, and asserting MBurstPrecise. MBurstLength, MBurstSeq and MBurstPrecise must be kept constant during the burst. MReqLast must be de-asserted until the last request in the burst. The timing diagram is shown is Figure 3-3.If the Slave IP is ready to accept one request, it captures the address of request and keeps SCmdAccept asserted. The Slave IP also responds to the request by driving DVA on SResp and the read data on SData. The slave must keep SRespLast de-asserted until the last response is finished.

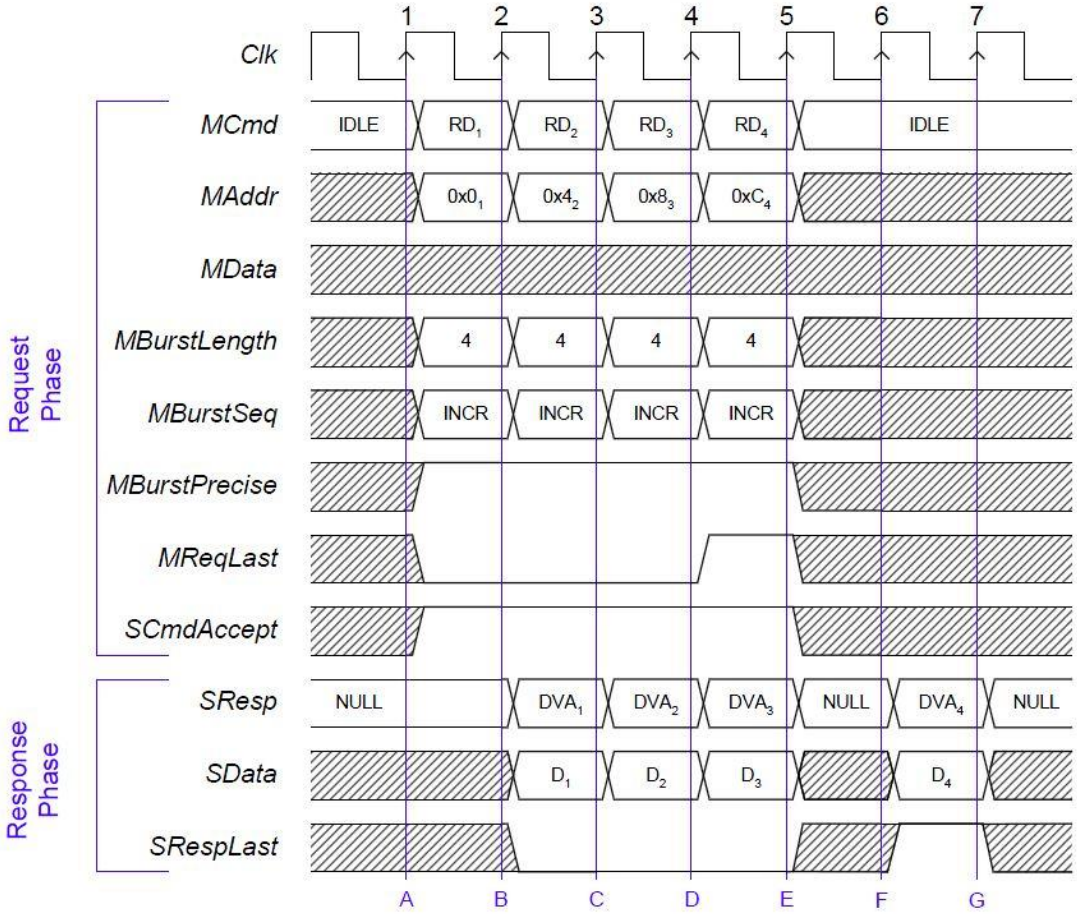


Figure 3-3 Timing diagram for burst read transaction (Source: [24])

3.3 Switching Technique and Routing Scheme

We use the most popular packet-based wormhole switching in a 4×4 mesh topology NoC. A packet is transmitted flit by flit and flows through the network. For regular 2-D mesh topology, the deterministic XY routing scheme is well suited for on chip communication [48-49]. XY routing is a dimension order routing, which routes packets first in horizontal direction to the correct column and then in vertical direction to the receiver. In the XY routing, five bi-directional ports in each router is encoded as a 3-bit binary code, which is shown as Figure 3-4. Each 3-bit code represents one hoc of current flit to the output port of router. In Figure 3-5, the XY routing path from

node A to node B is presented in red. The routing information is saved in routing tables and packet headers.

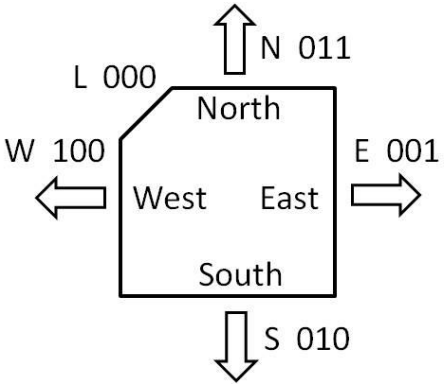


Figure 3-4 Direction encoding for five-port

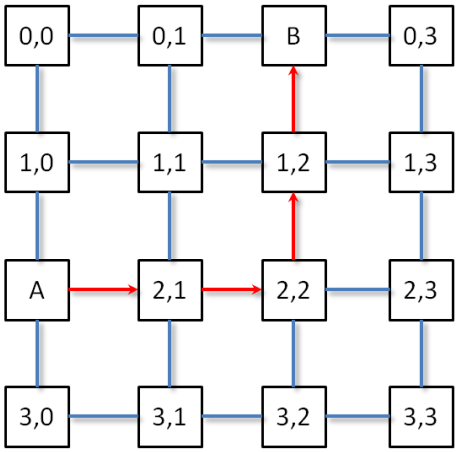
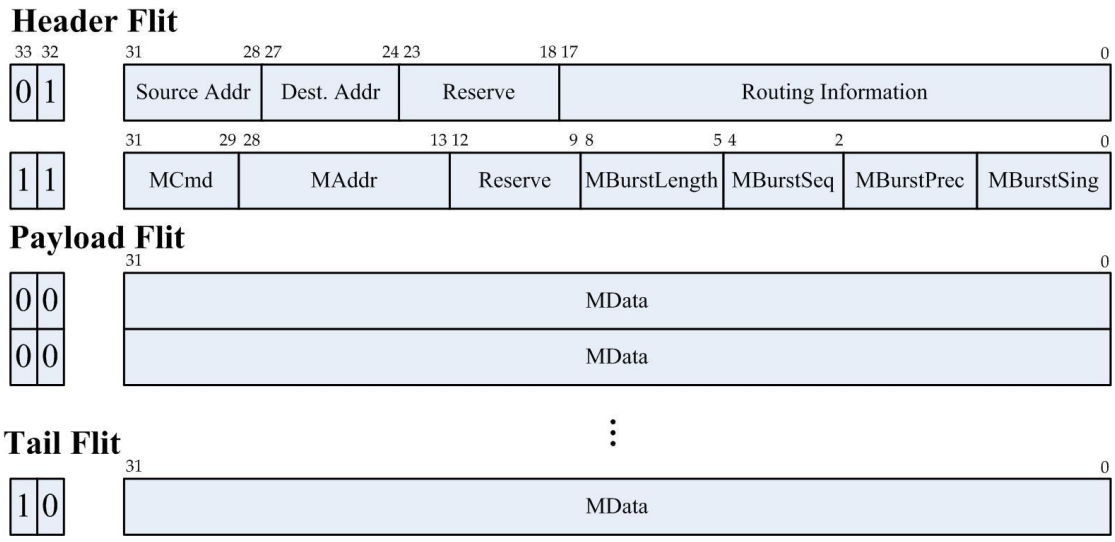
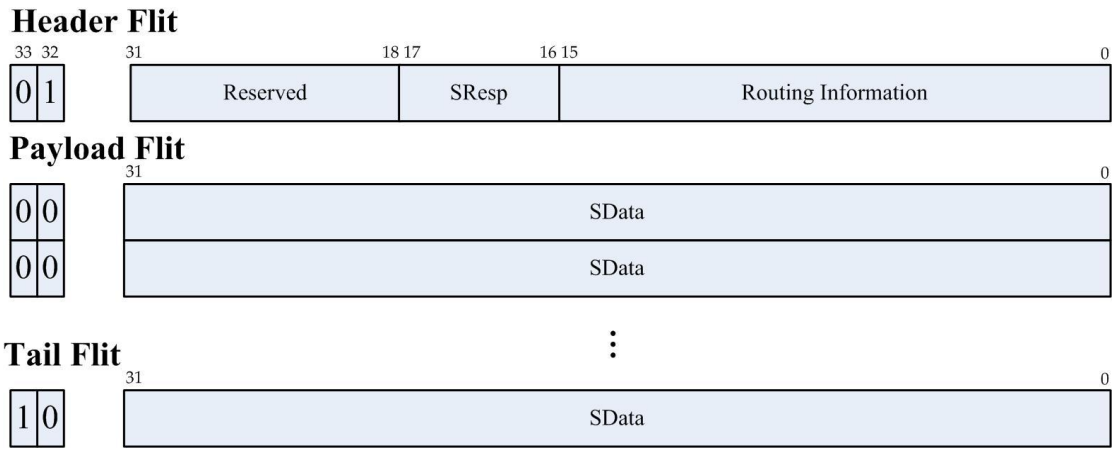


Figure 3-5 X-Y routing from node A to node B router

3.4 Packet Format



(a)



(b)

Figure 3-6 NI packet format (a) request packet (b) response packet

Packet format for NoCs tightly affects the NI control logic design, packet boundary scanning, and the area overhead of the NI. To reduce design complexity and balance cost and performance, we define two types of packets: request packets and response packets. The formats for these two types are shown in Figure 3-6. Request packet is always sent from a Master IP to a Slave IP and response packet is from a Slave IP to a Master IP. Each packet are divided into header, payload and tail flits. Two bits are used to present the flit type. Typically, a header flit contains control information and routing information. In our request packet, the header is composed of two flits. The control information encrypts OCP signals subset. The routing information presents route hops in XY-routing. If each hop is encoded with three bits, 18-bit routing information can support 6 hops in total, which is able to accommodate our 4×4 mesh topology. Reserved bits are for future extension.

3.5 Flow Control

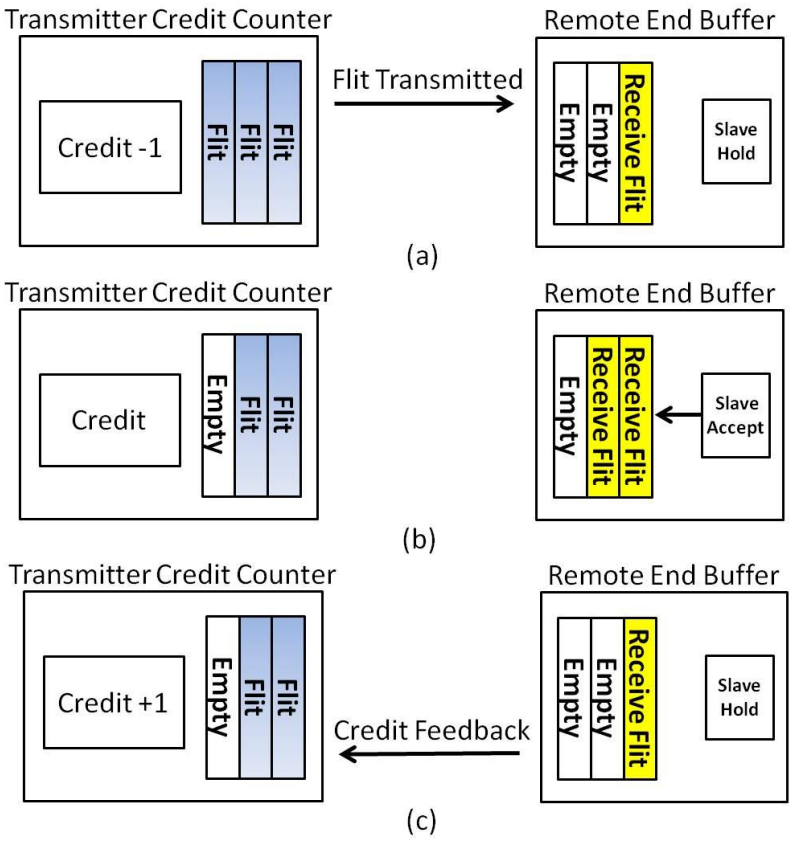


Figure 3-7 Credit-based end-to-end flow control mechanism (a) credit counter decrements when one flit is transmitted (b) slave IP core accepts received flit (c) credit counter increments when receiving feedback from slave IP

Initiator and Target NIs demand buffers to perform packet encapsulation and de-capsulation. Therefore, end to end flow control is necessary in NI to prevent buffer overflow in the receiver end, which could further lead NoC system crush. Credit-based and ACK/NACK are commonly used mechanism to avoid the buffer overflow at the destination node [25]. In the credit-based flow control, a credit counter is implemented in the transmitter to initialize the available buffer space at destination. The credit counter reduces one whenever the transmitter forwards a flit, shown as Figure 3-7 (a). If the count reaches zero, the buffer at the destination node

reaches the full status and thus no more flits are allowed. When one flit is consumed, the counter increases one. The credit is either piggyback in the response packet or provided by an extra link, shown as Figure 3-7 (c).

3.6 Initiator NI

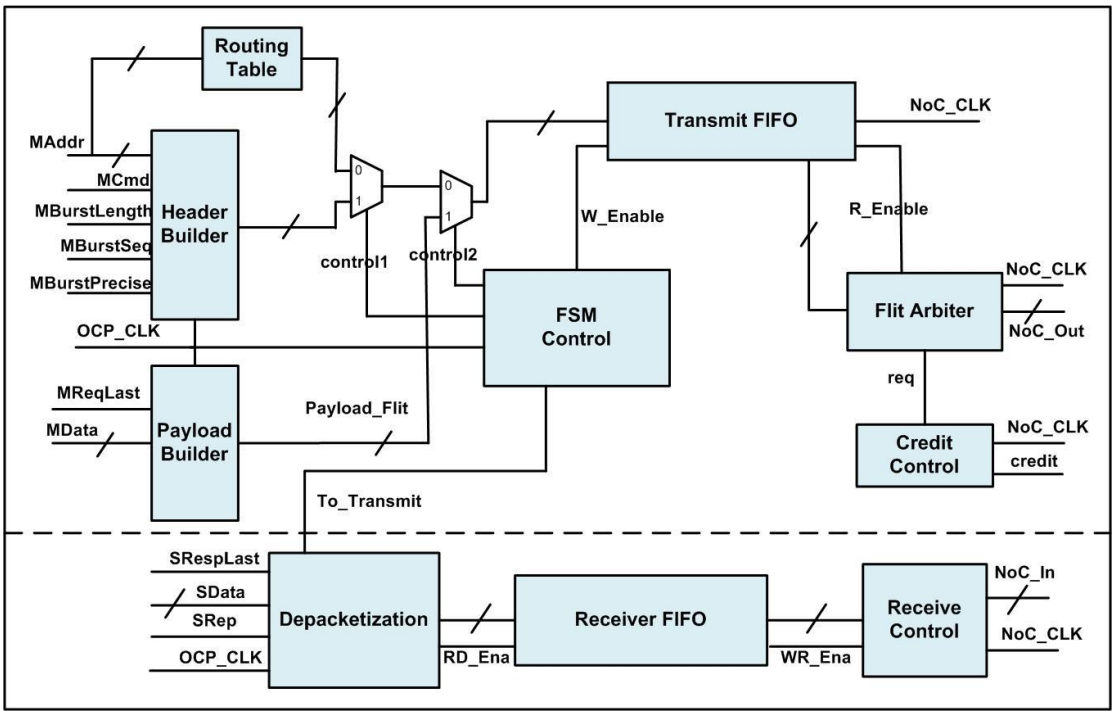


Figure 3-8 Block diagram of Initiator NI

The main tasks of Initiator NI are to receive the request from master IP core, encapsulate into a packet, transmit the packet flit by flit to the network, and receive response from the remote slave IP core. The proposed Initiator NI is shown in Figure 3-8. The transmitter is composed of header and payload builder, a routing table, asynchronous FIFOs, control logic and flit arbiters. The receiver consists of receiver FIFO, depacketizer and receive control unit. Among these sub-components, central FSM control, depacketizer and receive control are all FSM based control logic.

3.6.1 Header & Payload Builder

The header and payload builder mainly handle the handshake with the Master IP, accept and packetize the OCP signals presented by the Master IP. Particularly, payload header encapsulates the MData signal into payload and tail flits, and header builder encapsulates the rest of OCP signals into control information of the header flit. As soon as such process is done, the flits are ready to write into the transmitter FIFO. Figure 3-9 (a) and (b) present the gate level implementation of header and payload builder, respectively. The SCmdAccept comes from the NI FSM and will be asserted when the NI is ready to accept new transaction requests. The OCP signals except the MData will be registered by the header builder to form the header flit; and the MData will be registered by the payload builder to form the payload or tail flit. Tail flit is generated when the MReqLast from Master IP is asserted.

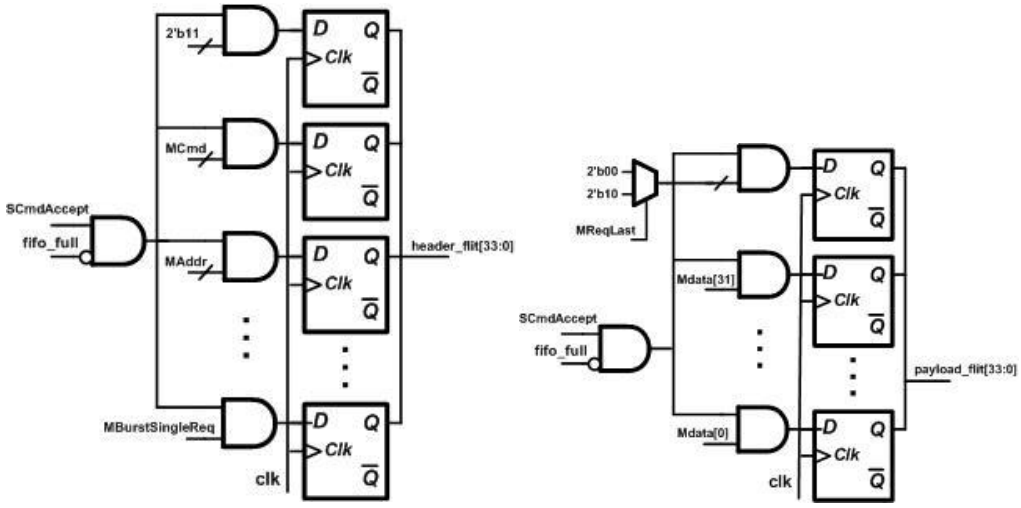


Figure 3-9 Packet builder (a) header flit builder (b) payload & tail flit builder

3.6.2 Routing Table

Routing table, or look-up table (LUT), is a local memory in Initiator to store the routing paths to other slaves within the NoC [31]. A routing table in NI is typically

implemented as a combination of content addressable memory (CAM) and random access memory (RAM), which is shown as Figure 3-10 [32, 33]. The global address of IP cores is written into the CAM and the routing path information is store in the RAM. When the MAddr is presented by the Master IP, CAM searches its contents to find a match and the routing path associated with global address will be retrieved from the routing table, encapsulated as a header flit and sent to the FIFO. In our design, the routing information is expressed as multiple 3-bit binary hocs.

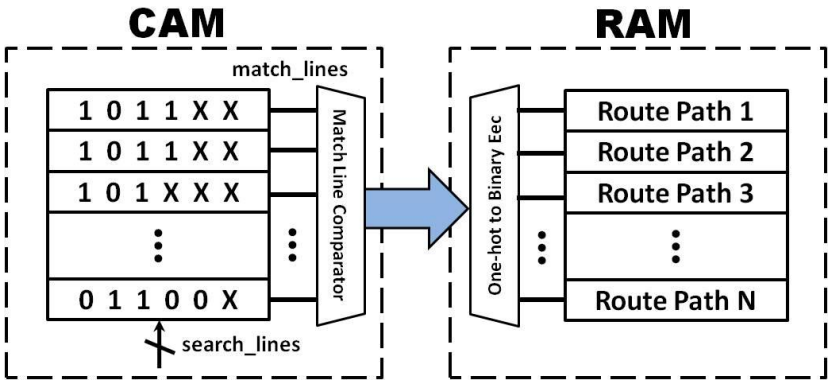


Figure 3-10 Conceptual view of a CAM-RAM based routing table

CAM is a special type memory for high speed searching and widely used in data compression, network switching, IP address filter and memory mapping [34-36].It can be either configurable or hard wired [37]. Unlike previous designs in [32-33], we refer to a configurable register-based CAM in our baseline architecture. Since a certain IP address may correspond to several different routing paths, the longest prefix matching method is utilized in a ternary CAM [34, 38]. As shown in Figure 3-11, the data_bits that store IP address and the search_bits are fed into the match line comparator bit by bit to check if they are identical. The care_bits signals are used to mask certain bits from the comparator comparison and a specific line will be selected if the longest

prefix bits are matched. The match lines output from CAM is in one-hot code, and therefore needs to be encoded as binary address before searching the routing path in the RAM.

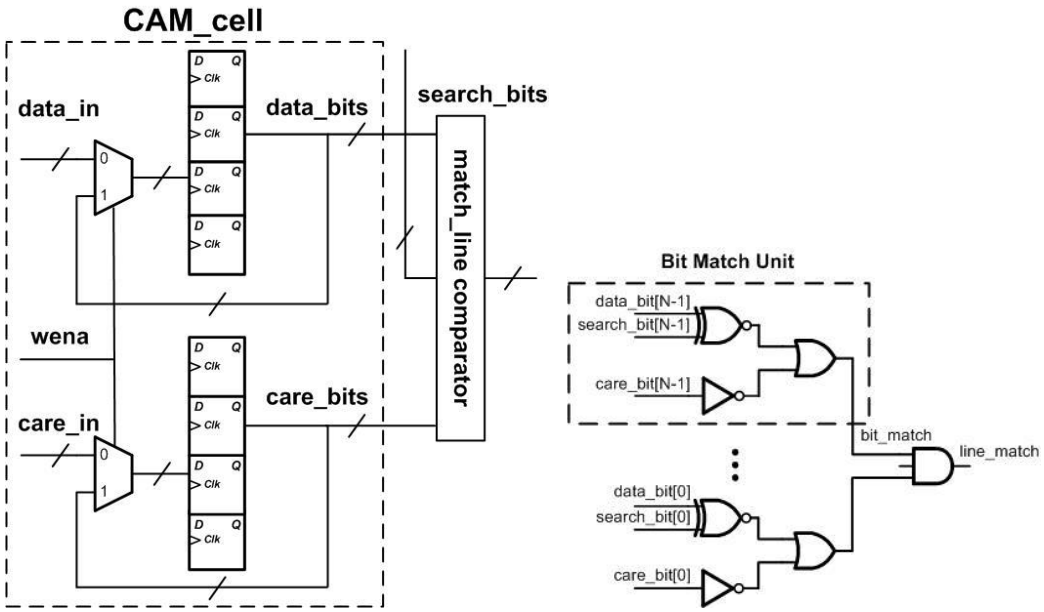


Figure 3-11 Longest prefix matching in ternary CAM: (a) single word cell unit of ternary CAM; (b) match line Comparator

3.6.3 Central Finite State Machine Control

The central FSM is the main control logic of transmitter in our proposed NI. The FSM is synchronous to the OCP clock, and is responsible for reordering and writing the flits into transmit FIFO after packetization. The details of state transition are presented in Figure 3-12. Since the Master IP may have write and read transactions, the FSM is therefore divided into two branches for the two types of transactions, respectively. In the write transaction branch, two header flits containing routing and control information and multiple payload flits are written into the asynchronous FIFO sequentially. The suspension states are designed to prevent writing when the FIFO is

full. In the read transaction branch, the header flits are written but there are no payload flits in read transaction. When the remote end provides response flits, the FSM will be reset to the idle state. Similarly, suspension states are design to prevent FIFO from reading when the FIFO is empty.

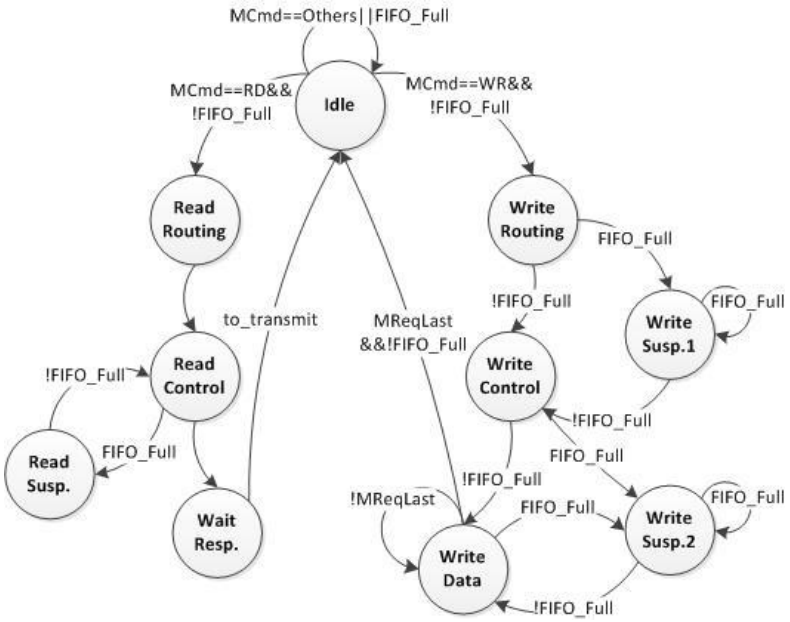


Figure 3-12 Central FSM control logic

3.6.4 Flit Arbiter

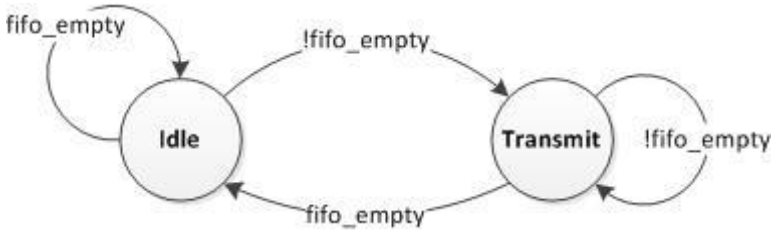


Figure 3-13 State diagram of flit arbiter

The flit arbiter is also a simple control logic but synchronous to the network clock. It receives the flits coming out of the transmit FIFO and sends flits to the NoC network. Figures 3-13 shows the detailed state transition. As long as the FIFO is not empty, flit arbiter will fetch and transmit the flits that are stored in FIFO. The credit

counter is also implemented with flit arbiter to realize the flow control described in 3.5.

3.6.5 Asynchronous FIFO

The OCP specific IP core and NoC network are typically operated at different clock frequencies, so an OCP clock and a network clock are defined for these two clock domains, respectively. Asynchronous FIFOs are therefore necessary in both transmitter and receiver paths for flit queuing, reordering and clock domain crossing.

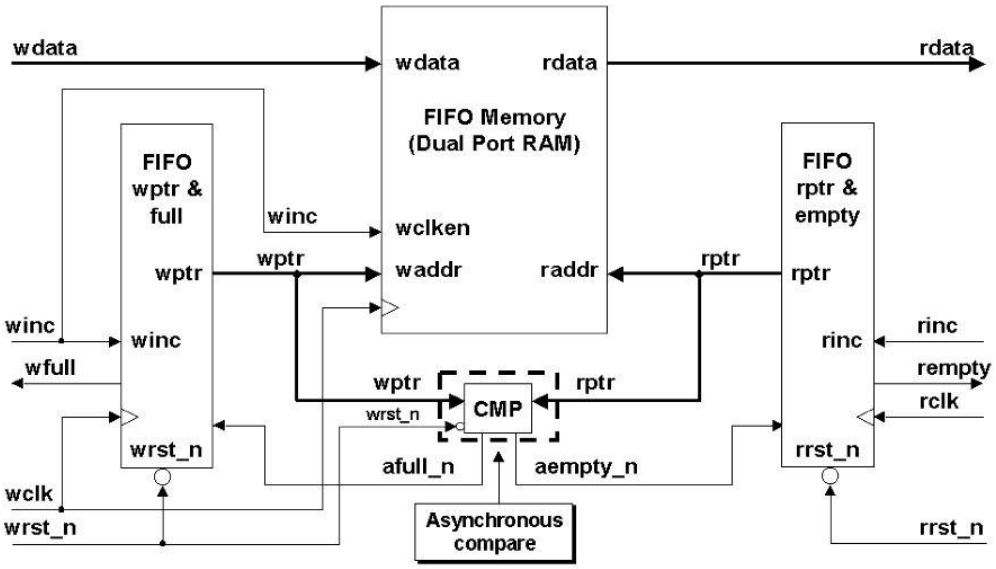


Figure 3-14 Dual-port RAM based FIFO architecture (Source: [39])

Dual-port RAM with asynchronous read and write pointers is a popular FIFO architecture, as shown in Figure 3-14. We adopt a similar design described in [40, 39] with binary and Gray code combined pointer to support high frequency (>250MHz).The FIFO depth is largely depended on the speed of both clock domains. Increasing the FIFO depth can help to accommodate longer size of packets and reduce the packet miss rate, but it also increases the power and area overhead [23, 41].

3.6.6 Depacketizer & Receive Control

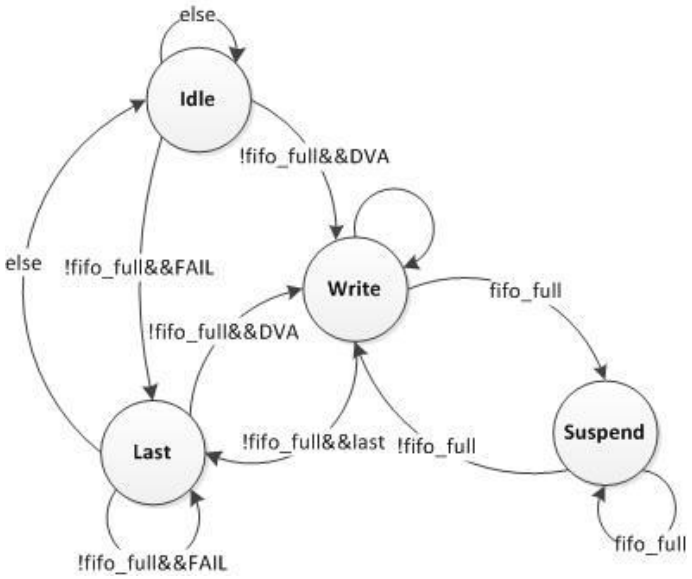


Figure 3-15 State diagram of receiver control

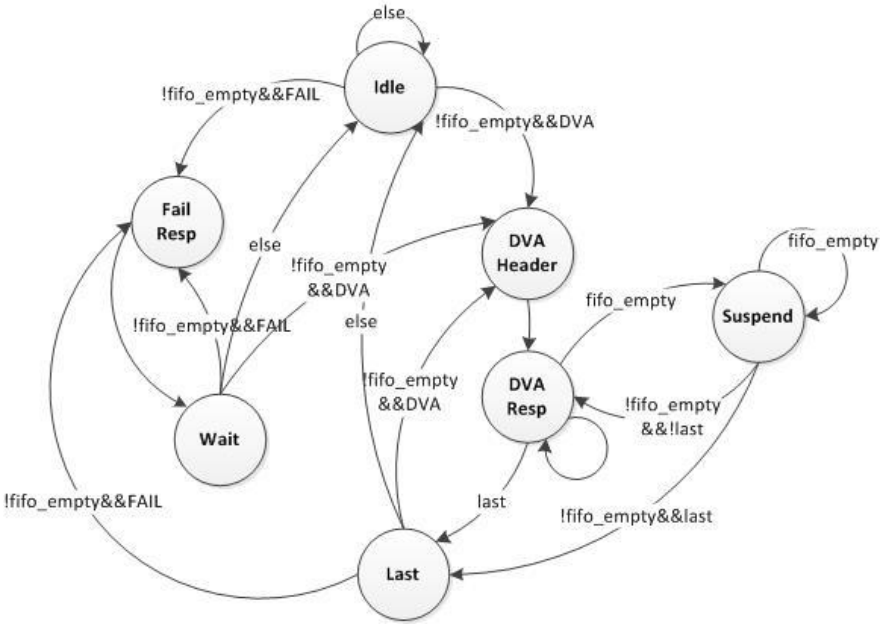


Figure 3-16 State diagram of depacketizer

The sub modules of receive control and depacketizer are control logic in the receiver path. The receiver path facilitates the reversed operation. The response flits from the Slave IP are synchronized with the receiver FIFO and then depacketized as handshake signals at the OCP interface. Since the flits from the Slave IP can be either

valid data or fail response, the control logics are also partitioned into two branches, as illustrated in Figure 3-15 and Figure 3-16, respectively.

3.7 Target NI

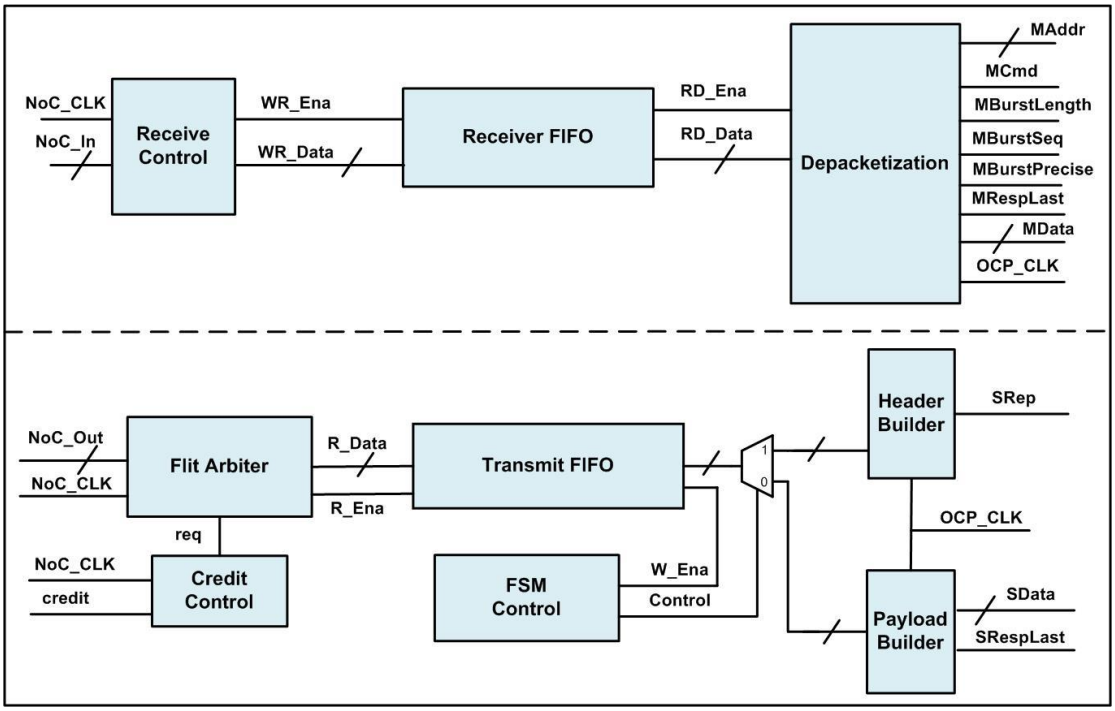


Figure 3-17 Block diagram of Target NI

Implementation of the Target NI (shown in Fig. 3.17) is opposite to that of Initiator NI in Figure 3-17. The main difference between target and initiator NIs is: the former one is an OCP master and the latter one is an OCP slave. The OCP signals asserted by the initiator NI (target NI) are complimentary to the target NI (initiator NI). Since Slave IPs can only passively receive requests, Master IPs' source address is retrieved from the request packet to provide routing information for response path. So Figure 3-17 of Target NI presents a mirrored architecture to the Initiator NI, which also

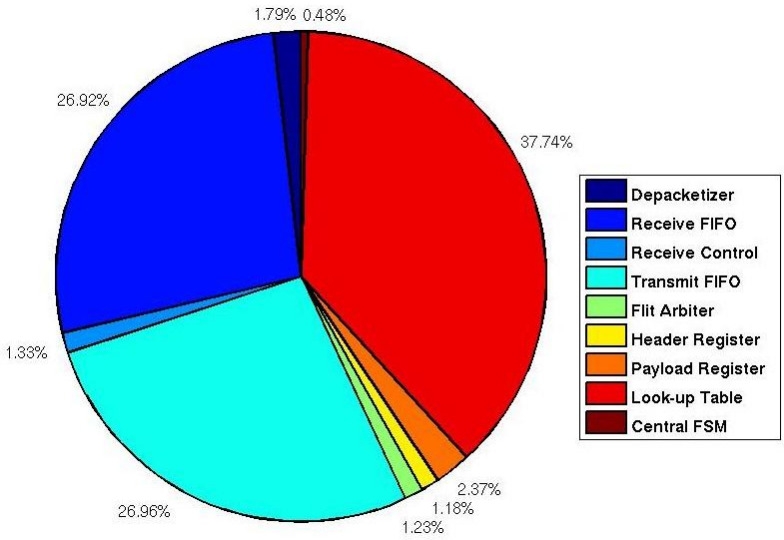
includes header/payload builder, asynchronous FIFO, flit arbiter, central FSM control, receive control logic and depacketizer.

3.8 Implementation and Simulation Results

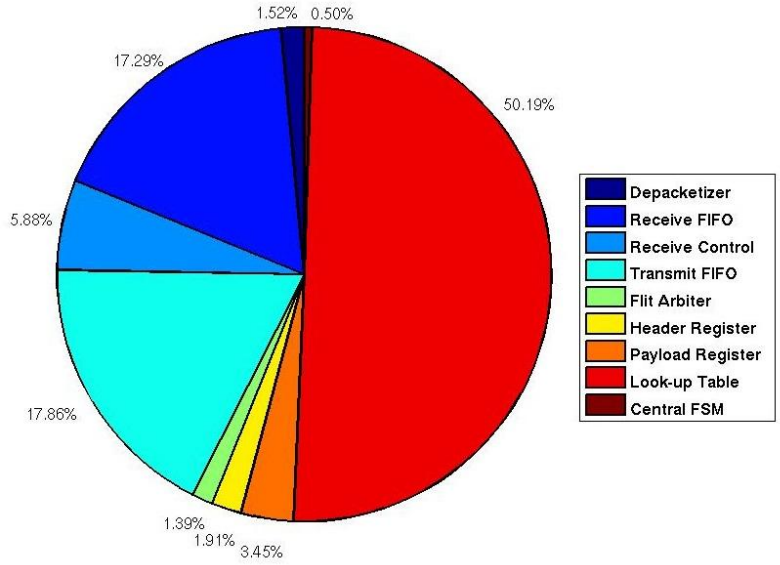
Table 3-3 Comparison of NI design in previous works

NO.	Freq. (MHz)	Area (mm^2)	Power (mW)	Latency (cycles)	Protocol	Tech. (μm)	Advanced Feature
[42]	312.5	0.43	N/A	[8, 10]	NA	0.13	CRC detection/ retransmission
[26]	725 /1086	0.058 /0.02	N/A	[4, 6]	OCP	0.13	GALS
[43]	500	0.036 /0.045	33.5 /36.9	[6, 10]	OCP	0.13	Basic
[44]	500	0.169	N/A	[4, 10]	OCP, AXI, DTL	0.13	Re-ordering, QoS
[45]	490	0.056	30.5	N/A	NA	0.18	Basic
[15]	500	0.141/0.172 0.166/0.172	N/A	N/A	OCP	0.13	Secure Memory Access
[12]	500	N/A	N/A	N/A	STBus	0.065	EMU/Security/QoS /Programmability etc.
This work	310	0.266/0.166	24.5	[6, 6]	OCP	0.18	Flow control/QoS

We implemented the proposed NI design with Verilog HDL and synthesized with Synopsys Design Compiler. Table 3-3 presents the general overview and comparison with other works. We adopt IBM 0.18μm CMOS-7RF technology to give a closer view. The maximum frequency, store-forward latency, area and power consumption were evaluated based on the synthesized netlist. Our implementation shows the comparable performance in terms of low cost and high speed with that in smaller-feature size technology.



(a)



(b)

Figure 3-18 Implementation details of different modules in NI (a) Area contribution to overall design (b) power contribution to overall design

Figure 3-18 (a) shows the complexity of each sub-module inside our NI, and Figure 3-18 (b) presents the contribution of each module to the overall power. We can conclude that the memory elements including asynchronous FIFO and LUT are the most costly inside a NI (over 80% in both cases). Our baseline design fixes the depth of asynchronous FIFO to16 support 16-flit long packet at maximum. As the FIFO

depth is highly parametric, it can be easily configured to a larger depth. Larger FIFO depth is more reliable and has smaller miss rate when NI operating at high frequencies, but it also increase the overhead [23]. Figure 3-19 shows such increasing trend on area versus the FIFO depth. Selecting an appropriate FIFO depth is a design tradeoff among speed, reliability and overhead.

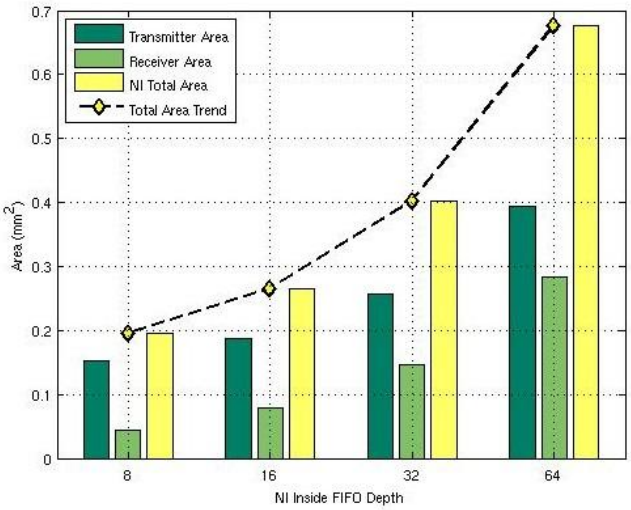


Figure 3-19 The increasing trend of NI area over FIFO depth

CHAPTER 4

HARDWARE TROJAN ATTACK MODEL

The hardware attack framework or threat model to NoC was first discussed in [11, 26-27]. HT attacks in NoCs can be generally categorized into three categories as follows.

- *Denial of Service*: A DoS attack attempts to degrade system performance. The adversary may frequently send malicious requests to the victim of attack, resulting in extra traffic congestions and the increase on network transmission latency. More precisely, HTs can cause bandwidth reduction, incorrect routing path, livelock and deadlock.
- *Extract of Secret Information*: Secret information extraction means that HT facilitates hackers to steal sensitive or critical information from authorized memory or registers, such as the crypto keys. With the assistance from HTs, secret information can be extracted by unauthorized read request, buffer overflow or duplicating read operation to transmit data to an un-trusted destination.
- *Hijacking*: Hijacking refers to altering the regular execution flow or modifying the system configuration so that specific tasks set by the attacker can be executed. One example for Hijacking is to exploit buffer overflow to bypass the digital serial code in video games.

The tangible security attack examples provided in [11, 26-27] are either not thorough or in conceptual or abstract level. Other security discussions reported in [28,

30-32] either only study the attacks on memory blocks or assumes that the attack come from external source. In fact, the HTs can also be inserted in NoC infrastructure as well. In this section, we summarize a practical HT design approach and attack model specifically for NI. Particularly, we analyze and propose multiple rare switching nodes which can potentially be the HT trigger. Several meaningful HT payload locations are also presented and evaluated.

4.1 HT Models in Previous Designs

DARPA issued its first call for the study of hardware Trojan. In [15-16, 63-64], different methods were proposed to classify HTs based on various characteristics, such as insertion phase, trigger mechanism, payload effect and location. HT can also be either digital or analog [8]. Since the NoC infrastructure is mainly a synthesizable digital design, here we only limit our discussion to digital HTs.

A basic HT model consists of two parts: (a) an activation mechanism referred as HT trigger; (b) an intrusive circuit referred as HT payload to affect original design [65-66].HT payload can be simple logic gate such as XOR to flip or change the logic value of original logic. However, the HT trigger is more complicated since the HT needs to be triggered under a certain condition or over a long period of time. Several examples of HT are shown in Figure 4-1 [8, 46].

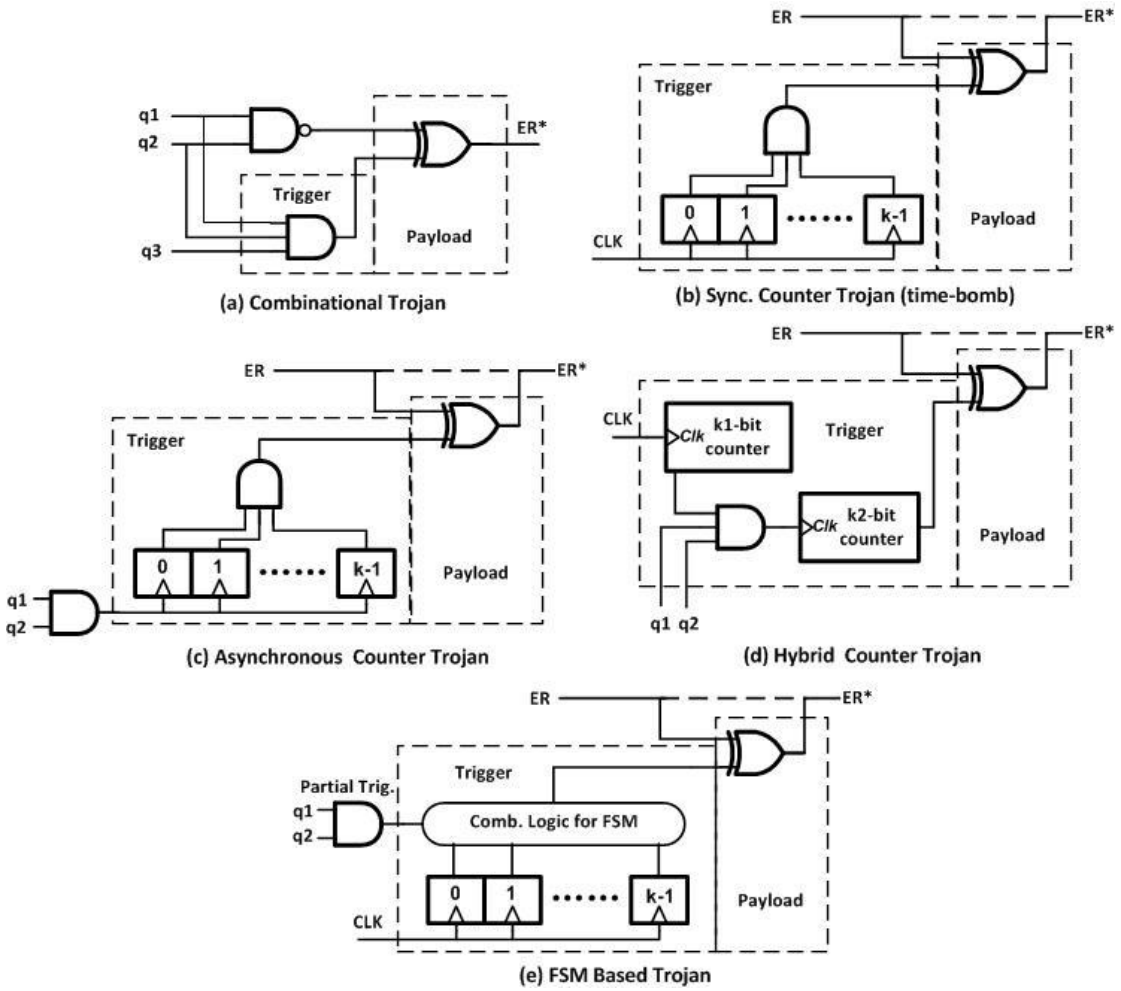


Figure 4-1 Examples of HT with various triggers (a) combinational (b) synchronous counter (c) asynchronous counter (d) hybrid (e) FSM based

Digitally triggered HTs can be again classified into combinational and sequential types. Combinational Trojans shown as in Figure 4-1 (a) are activated by the occurrence of rare logic value combinations. The occurrence of the condition $q1 = q2 = q3 = 1$ at the trigger nodes causes the bit flip at node ER^*. Sequentially triggered HT, on the other hand, are activated by the occurrence of a sequence or a period of continuous operation. The free running synchronous trigger in Figure 4-1 (b) is the simplest example. The Trojan will be triggered when the counter counts to 2^kclock cycles working as a time-bomb. The clock in Figure 4-1 (b) can also be

replaced by a logic gate to make it an asynchronous counter Trojan as shown in Figure 4-1 (c). The counter will only increment when the combination $q1 = q2 = 1$ is met. Hybrid counter Trojan in Figure 4.1 (d) combines the features of combinational and sequential Trojans. More complex state machine Trojan, shown in Figure 4.1(e), is also recently discussed in [47-48]. The Trojan output is activated only when reaching the last Trojan state.

4.2 HT Designs

The examples discussed in 4.1 widely exist in the HT design of various digital applications, such as general purpose processor [18, 69], cryptographic IP [70-72], memory [73-74] and communication interfaces [75-76]. These can be configurable IP cores within a NoC system, but unfortunately, none of them touch upon the HT design in NoC infrastructure. Further, one obvious shortcoming in previous work is that very few of the authors consider a practical Trojans that would be able to avoid traditional detection such as functional verification, code coverage and post-silicon testing [48, 49]. For example, the free running synchronous counter Trojan requires large area/power overhead to guarantee a certain trigger time, which is not feasible in a high speed networking system. Also, most of the designers choose randomly locate HTs in the chip and were not able to perform meaningful attacks. So we summarize the key features of a practical HT design in NI.

- Maintain main functionalities of original circuit when it's not activated
- Create minimum overhead in terms of power, area and delay

- Locate in key points in the circuit to perform meaningful attacks
- Able to remain dormant in traditional detection techniques such as functional verification, code coverage and post-silicon testing

4.2.1 HT Triggers in NI

The key idea of HT trigger design is to realize the significant low trigger probability with minimum logics. Among the HT trigger examples in 4.1, FSM based sequential Trojans have been proven to be extremely stealthy in nature in most recent works [17, 68,77].The trigger probability can be exponentially lower by increasing the length of rare trigger events. Another advantage of FSM HT is that it can be embedded into the existing unused states of the FSM in original circuit. NI is a module with numerous FSM based control logics, which creates lots of opportunities for HT insertion. We expand the discussion in [17, 68,77] and specify several cases of HT shown as in Figure 4-2.

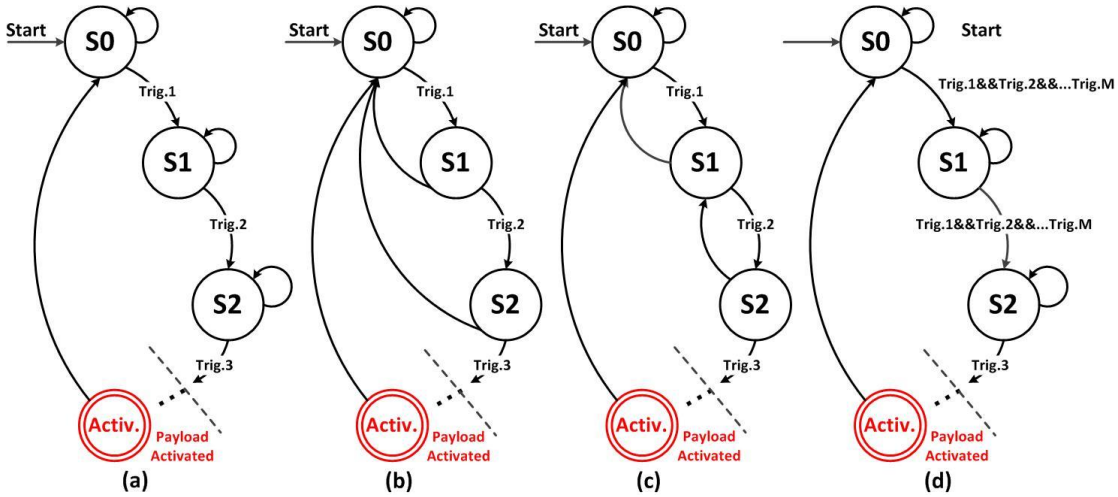


Figure 4-2 Various scenarios of FSM based HT insertion

The state transition condition of FSM depends on the rare event or rare switching

node in original circuit, and therefore does not cause extra overhead. Figure 4-2 (a) passes through a series of intermediate states$S_0 \dots S_N$ before activation. Figure 4-2 (b) and (c) are slightly different. They will transmit backwards if the trigger condition is not met. Figure 4-2 (d) requires all the trigger conditions to be met at each transition.

In [17, 68,77], the authors also provide an approach to estimate the probability of HT trigger and the expected activation time in terms of clock cycles. Let p_i $(1 \leq i \leq N+1)$ denote the probability of the Trojan transition from state S_{i-1} to S_i, where $N = 2^k - 2$ and k is the number of D flip-flops used in trigger, the trigger probability and activation time can be roughly simulated using a Markov process. Table 4-1 gives a summary of HT triggers discussed and their feasibility to NI. M denotes the number of rare event used in NI HT design and the trigger probability can be roughly estimated using Markov chain. If the multiple trigger signals are independent events, the activation time can be estimated using the reciprocal of trigger probability.

Table 4-1 Summary of various HT triggers and feasibility in NI

HT Type	Trigger Probability	Activation Time (cycle)	Feasibility to NI
Comb. Trojan	$\prod_{j=1}^{M} p_j$	$\frac{1}{\prod_{j=1}^{M} p_j}$	Applicable
Sync. Counter	100%	2^k	Not Applicable
Async. Counter	$(\prod_{j=1}^{M} p_j)^{2^k}$	$\frac{2^k}{\prod_{j=1}^{M} p_j}$	Applicable
Hybrid Counter	$(\frac{1}{2^{k1}} \times \prod_{j=1}^{M} p_j)^{2^{k2}}$	$\frac{2^{k1+k2}}{\prod_{j=1}^{M} p_j}$	Applicable
FSM (a)	$\prod_{i=1}^{N+1} p_i$	$\sum_{i=1}^{N+1} \frac{1}{p_i}$	Preferred

FSM (b)	$\prod_{i=1}^{N+1} p_i$	$\frac{1}{\prod_{i=1}^{N+1} p_i}$	Preferred
FSM (c)	$\prod_{i=1}^{N+1} p_i$	$\frac{1}{\prod_{i=1}^{N+1} p_i}$	Preferred
FSM (d)	$\prod_{j=1}^{N+1} \prod_{i=1}^{M} p_i$	$\frac{1}{\prod_{j=1}^{N+1} \prod_{i=1}^{M} p_i}$	Preferred

Besides the HT trigger models discussed in 4.1, we can conclude that various types of FSM trigger are also perfectly feasible in NI. The number of such sequential trigger conditions can be unmanageable large by applying different trigger sequences, which help the Trojan hidden in conventional detection. If the attackers are intelligent enough, the trigger nodes can be carefully picked from the original NI circuit, which will be further discussed in 4.3.

4.2.2 HT Payloads in NI

We summarize several types of payload circuits in terms of their effects in NI, which are shown in Figure 4-3. They can be single gate or gate array applied to multiple bits of the intruded signals. XOR type payload in Figure 4.3 (a) typically flips the logic value of original signal; AND type payload in (b) masks the signal when trigger asserts; dually, OR type payload can force victim signal to logic 1 in (c); MUX type payload in (d) can replace the target circuit with pre-set values and DeMUX type payload can extract security sensitive information to eavesdropper. The payload locations mainly are carefully chosen to perform meaningful attack and largely depend on the objective of attacker. The details will be cover in Section 4.4.

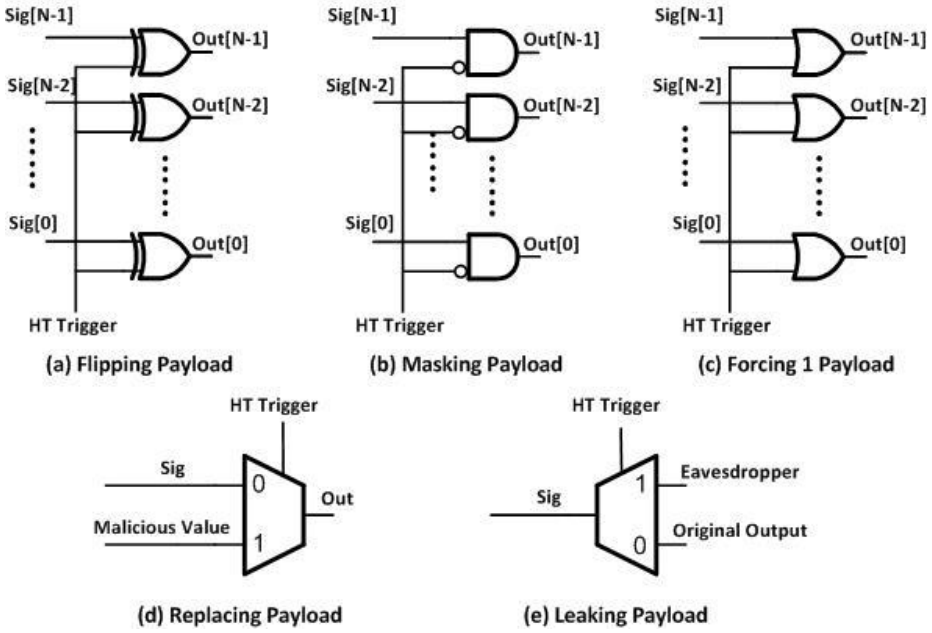


Figure 4-3 Examples of various HT payloads

4.3 Potential Trigger Signals for HTs in NIs

4.3.3 Potential Trigger Signal 1: Reset Signal

In the NI, the *Reset* signal is typically required to clear storage elements; more specifically, they are header & payload registers, asynchronous FIFO, routing table, and state registers of FSM. Like many other digital systems, the transition frequency of the reset signal is extremely low; thus this signal has a potential to be utilized in the HT trigger circuit. The reset-based HTs can easily escape from the HT examination tests, if HT designers use the reset signal as an input for a sequential HT, which is not triggered at the first transition time of the reset signal.

4.3.4 Potential Trigger Signal2: Unused States in an FSM

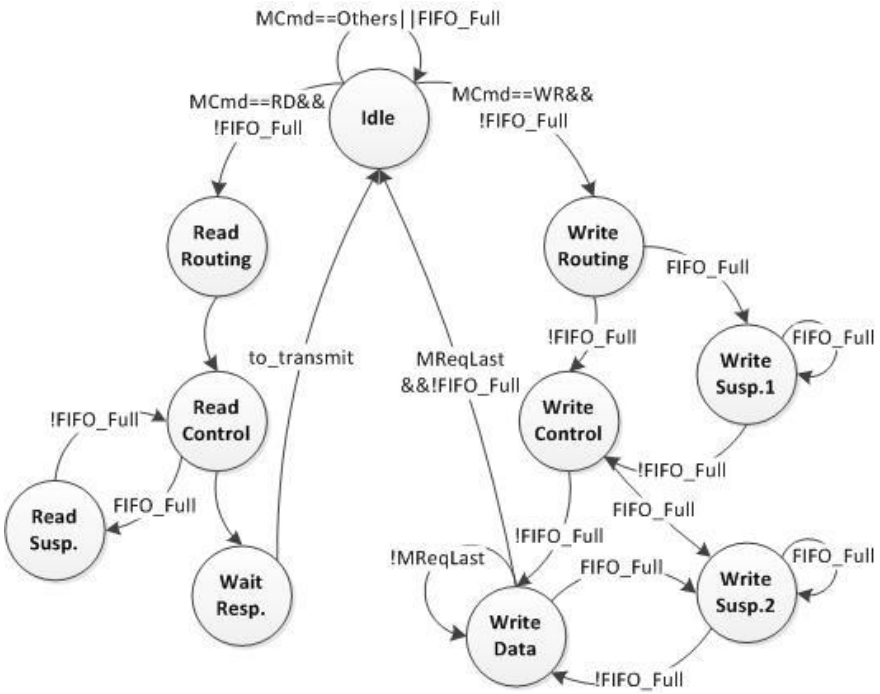


Figure 4-4 FSM for initiator NI. Signals on arches are OCP signals and FIFO full indicator

In NI, a finite state machine (FSM) is typically needed to coordinate packet read and write operation. Figure 4-4 shows the central FSM for a NI compatible with OCP protocol. That FSM is composed of ten legal states, thus requiring four bits to represent the states. As four bits can represent 16 states in total, six states will remain as unused. Unused states in a FSM are common for a complicated control system. To minimize hardware cost, such unused states are typically remarked as unspecified state in logic optimization. Consequently, those unused states could be utilized by a hacker as a HT trigger. Ideally, the state transition only happens within the legal states. However, under certain circumstance such as voltage drop, crosstalk and substrate noise, the FSM may switch to an unused state.

Although most of the FSMs have self-recovery and protection mechanism, the

transition to unused states is hard to be prevented completely. If a HT trigger circuit takes advantage of such low-probability events, the HT effect cannot be easily discovered by the functional verification.

One example of unused state transition can be caused by soft errors. In a k-bit FSM, we assume that only M out of 2^k possible states are in use. The soft error rate is in the range of $10^{-8}\sim10^{-12}$. If one FSM state bit is flipped by a soft error, the probability of illegal transition due to soft error can be roughly estimated in equation (4-1).

$$P_{un} = (10^{-8}\sim10^{-12}) \times (1 - \frac{M-1}{2^k}) \quad (4\text{-}1)$$

4.3.5 Potential Trigger Signal 3: FIFO-Full Signal

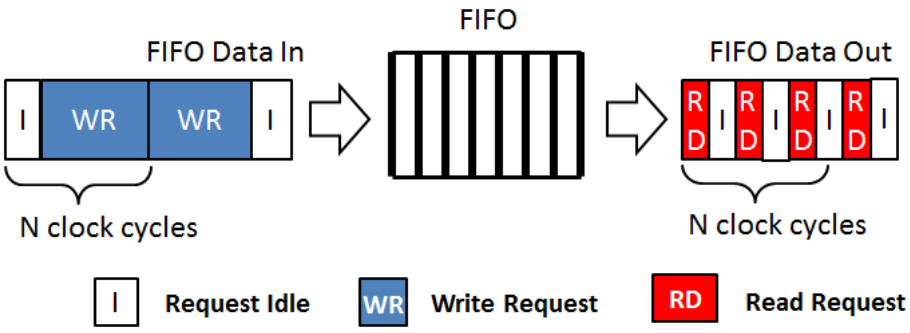


Figure 4-5 FIFO full caused by writing burst

FIFO buffers are essential components for data storage in NoC design. In NI, FIFO is used to queue packets, as the IP core clock frequency is different with the NoC clock frequency. The FIFO depth is a tradeoff between hardware cost and NoC performance [40]. Increasing the FIFO size can optimize the NoC latency and throughput at the cost of increased chip area and power consumption.

As the write and read operations may execute at different frequencies, the speed of

write-in operation should be no more than that of read-out operation; otherwise the FIFO will be always full. Let's use f_w and f_r todenote the clock frequencies for write and read operations, respectively, and use λ_1 and λ_2 to represent the write-in and read-out data rates, respectively. To avoid FIFO full, one should design a FIFO that meet the requirement expressed in equation (4-2).

$$\mathrm{f_w} \times \lambda_1 \leq \mathrm{f_r} \times \lambda_2 \qquad (4\text{-}2)$$

However, the constraint in equation (4) only works well for average cases. If writing burst happens and the FIFO depth is not large enough, the read operation with the average read-out rate $\lambda_2$will cause FIFO full. As shown in Figure 4-5, in 2*N*clock cycles, there are $2\mathrm{N} \times \lambda_1$ consecutive write operations. However, during the time period of $2\lambda_1 \times \frac{1}{\mathrm{f_w}}$, there are only $2\lambda_1 \times \frac{1}{\mathrm{f_w}} \times \mathrm{f_r} \times \lambda_2$read operations. As a result, the minimum FIFO depth for NI should be equal to the expression in (4-3).

$$\text{FIFO Depth} = \mathrm{N}(2\lambda_1 - 2\lambda_1 \times \frac{1}{\mathrm{f_w}} \times \mathrm{f_r} \times \lambda_2) \qquad (4\text{-}3)$$

If the FIFO depth in the NI implementation is less than the value indicated in equation (4-3), the FIFO will be full occasionally. This is because the requirement in (4-2) is for average cases. In the specification stage, λ_1 and $\lambda_2$are obtained based on infinite time. In reality, the peak value of the read and write speed may temporally exceed the defined data rates for a short period of time. Such temporal event could be used in HT trigger design.

4.4 Potential Payload Locations in NI

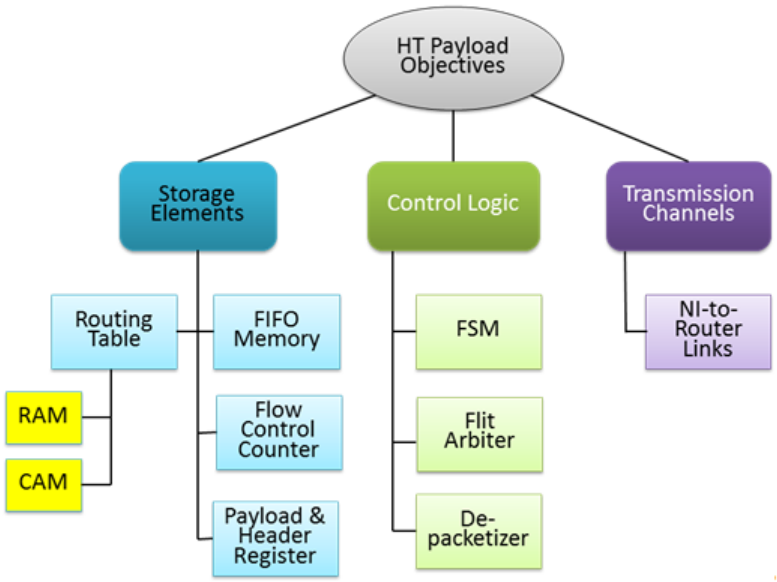


Figure 4-6 HT payload objectives on NoC network interface

We classify the HT payload objectives into three categories: storage elements, computation or flow control units, and transmission channels. The procedure to change the content in each category is slightly different. The influence of the HT payload location in each category is different, as well.

Figure 4-6 shows possible HT payload injection locations on a NoC NI. In the storage elements, similar to router, one HT payload can change the FIFO content, such as flit type, source/destination node address. Changes in the routing table can manipulate the list of trusted IP cores and secure routing paths. The protocol specific signals can be changed in the payload and header building registers. HT invasion on these storage elements can authorize the access to restricted memory area and thus cause secret information leakage and hijacking. HT influence on the control logic such as FSM and de-packetizer can disorder the flit sequence in a packet, which

further causes system FSM remaining in the same state or even be crashed.

4.4.1 Payload Objective 1: Latching Global Clock

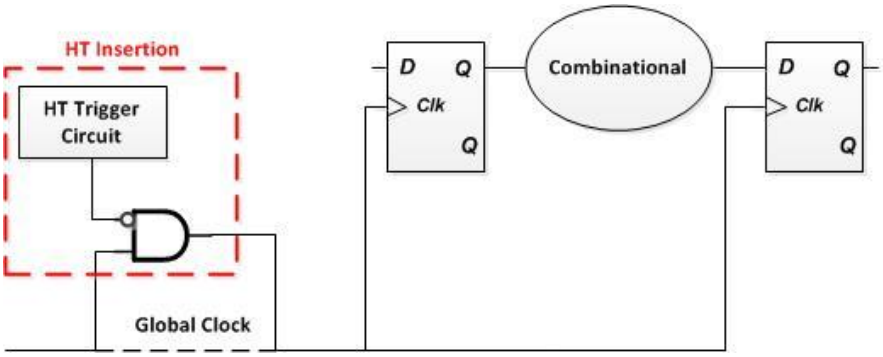


Figure 4-7 HT payload at clock tree circuitry

The clock distribution network can be regarded as the most critical module in a digital system. It provides global clock source to rest of the circuitry for synchronization and in most cases, several local clocks are derived from the global clock through a PLL. Shown as in Figure4-7, if the HT attack happens at clock circuitry by latching or freezing it up, the entire chip does not function unless the clock is re-enabled [46]. It may also possibly cause glitches when the HT trigger transits from dormant to active. Either way will cause severe impact to the rest of system. This Trojan can be inserted at design as well as fabrication phase and can be described at gate level.

4.4.2 Payload Objective 2: Damaging Flit Type Information

Bit stream transmitted over the NoC fabric is organized with a certain format, i.e. packet. One packet is typically composed of one (or two) header flit(s), several payload flits and one tail flit. Figure 4-8 shows one popular packet format. Other formats can be found in [41-43]. No matter what format is used in packetization, a

common feature in all packet formats is a few bits to indicate the flit type: header, payload, or tail. Two, at least, reserved positions on each flit are used to differentiate the flit types, as the router relies on these two bits to determine whether and how to compute the next routing path or whether the current routing channel should be released.

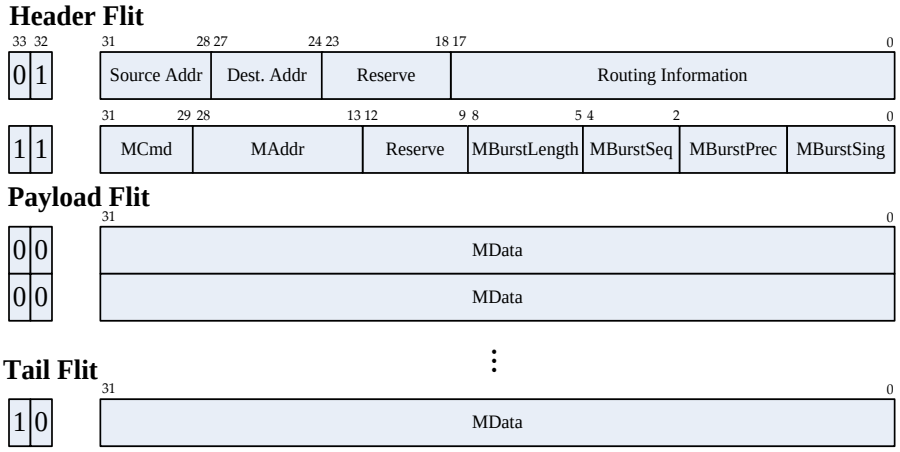


Figure 4-8 Transmit packet format in proposed NI design

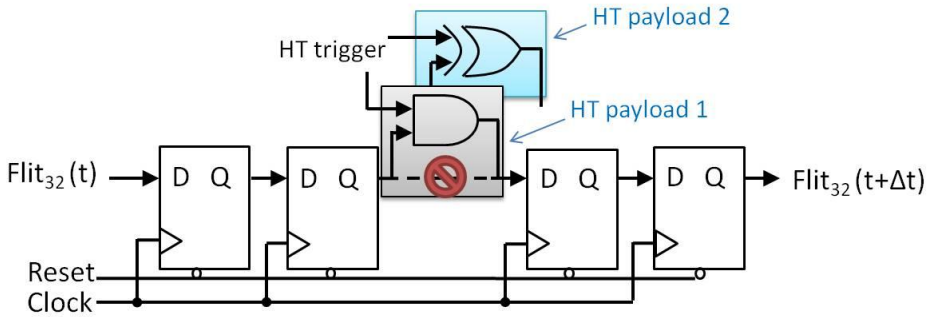


Figure 4-9 HT payload at FIFO register chain

The HT payload circuit to damage the flit type bits can be any circuit that leads to a stuck-at-0/1 or logic flipping error. HT payloads 1 and 2 shown in Fig. 4-9 are examples for stuck-at and logic flipping payload circuits, respectively. If the 32nd flit bit is muted by an active HT inserted in the FIFO register chain, the header flit cannot be recognized by router, thus leading to a packet loss. We propose a pseudo-code

based HT attack model in Figure 4-10. The key idea of this model is to modify the flit bits representing the flip type. The logic flipping can be executed at gate level or register transfer level.

```
//Pseudo code for the HT attack model of losing flit type information
if(HT trigger condition is true)
    HT_payload <= on;
else
    HT payload <= off;
//HT injection location
if(HT payload == on)
case(loss type):
            Header lost:
flit[32]<=and(flit[32],!HT payload) at time Tj
            Tail lost:
flit[33]<=and(flit[33],!HT payload) at time Tj
            Packet fission:
flit[32]<=xor(flit[32], HT payload); at time Tj?j+pktlen
    flit[33]<=xor(flit[33], HT payload); at time Tj?j+pktlen
endcase
else
    Execute normal operations;
```

Figure 4-10Pseudo code for HT damaging flit type information

4.4.3 Payload Objective 3: Altering Routing Path

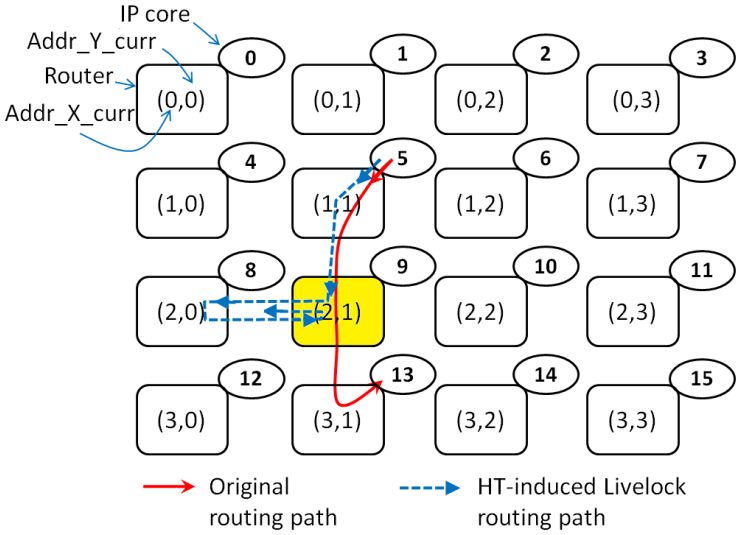


Figure 4-11 An example caused by an HT payload on the node current address. Solid red line is original routing path. Dashed blue line is livelock path caused by a HT that changes router (2, 1) current address to (2, 2)

Malicious insertion can intentionally change the routing path either routing table

for adaptive routing to cause deadlock or livelock, thus wasting system resources and degrading NoC performance.

- *Livelock Example*

Livelock means the packet is traveling through network but does not get any close to the destination. Livelock is typically caused by illegal routing turns. Let's use the most popular routing algorithm, XY routing, to introduce the possible way to introduce a HT payload that causes illegal turns. The router current address (*Addr_X/Y_curr*) is hardwired for XY routing path computation. In XY routing, a packet first goes through the hops on the X direction, and then goes through the hops on the Y direction. As the hard wired signal does not have switching activities, the modification by rarely-trigged HT on those signals is hard to be detected by traditional test approaches and code coverage analysis methods.

We use a mesh NoC with 16 nodes (i.e. 16 IP cores) to explain how a livelock happens if an HT alters the current address. The livelock example is depicted in Figure 4-11. Assume IP core 5 attempts to send a packet *Px* to IP core 13, using XY routing. The original routing path for *Px* is starting from router 5, through router 9 and ending at router 13. The two-dimension representation for those three routers are (1,1), (2,1) and (3,1), respectively. If a trigged HT changes the hardwired current address of router (2,1) to a new address (2,2), now the packet *Px* will go through the west output port of router (2,2) and reach the east input port of router (2,0). As the destination of *Px* is IP core 13, *Px* is directed back to router (2,1) based on the rule of XY routing

algorithm. However, the current address of router (2,1) is maliciously hardwired as (2,2), the packet *Px i*s transmitted back to router (2,0) again. Consequently, the packet *Px* comes back and forth between router (2,0) and router (2,1) until the HT effect is gone; packet retransmission after using time-out mechanism does not help to resolve the problem.

```
//Pseudo code for the HT attack model of altering packet routing path
if(HT trigger condition is true)
    HT payload <= on;
else
    HT_payload <= off;
//HT injection location
if(HT payload == on)
case(Location):
//----------before route computation-----------------
        Routing table:
//Modify content in CAM for routing calculation
             CAM[legal access request]<= untrusted IP core Address;
        Flit FIFO:
//Modify packet destination address bits in header flit
            FIFO[wr/rd_pointer][27:24]<= untrusted IP core Address;
        R-R links:
//Bit flip dest. address when header flit passes over the links.
            Link[link ID][27:24]<= untrusted IP core Address;

endcase
else
    Execute normal operations;
```

Figure 4-12 Pseudo code for HT attack model: altering packet routing path

We propose a HT attack model that alters the routing path in NoC. The pseudo code for this model is shown in Figure 4-12. We list the potential locations for HT insertion and the possible way to modify the content or logic in the vulnerable locations. The ways we illustrated in Figure 4-12 are just typical examples. In reality, there will be many equivalent approaches to perform similar attacks to alter packet routing path.

4.4.4 Payload Objective 4: Injecting Redundant Packets to Increase Traffic Congestion

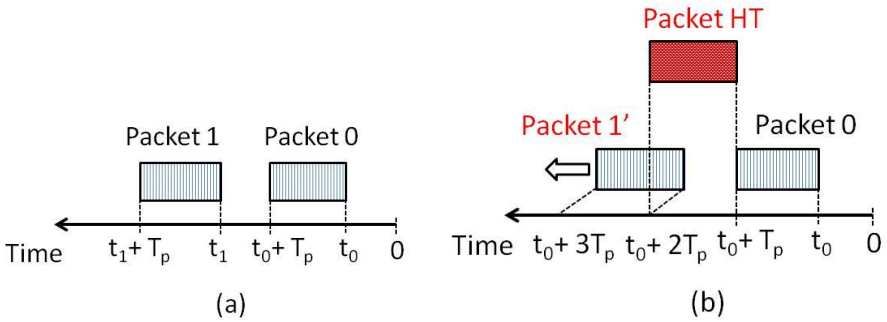


Figure 4-13 Redundant packet injection caused by a triggered HT in NI (a) Normal packet injection from an IP core (b) A redundant packet injection between two valid packets by a HT

Livelock and deadlock interrupt the NoC normal operation. There is another HT attack that purely degrades system performance. In NI, the triggered HT can maliciously force one NI duplicating every packet it submitted. If the consecutive packet arrives before the end of the duplicated packet, the starting time for that consecutive packet will be delayed, as shown in Figure 4-13.

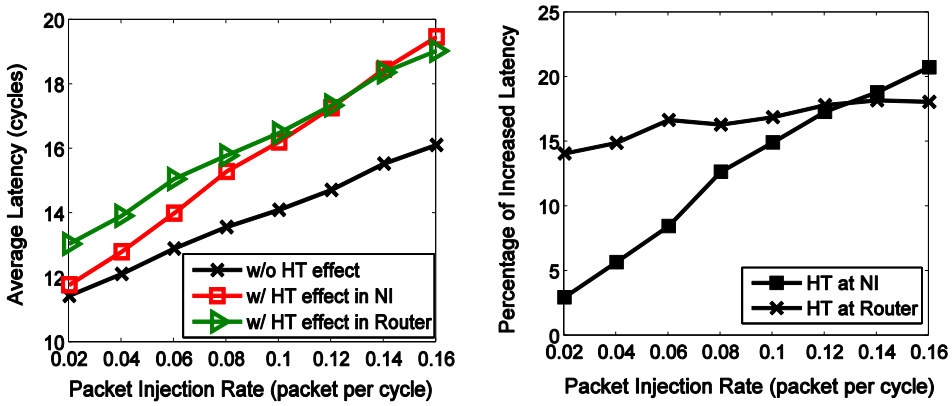


Figure 4-14 The impact of HT on latency (a) Average packet latency comparison for w/ and w/o HT induced redundant packet injection (b) Percentage of increased average packet latency by HTs

We conducted an experiment to quantitatively compare the performance

degradation caused by HT-induced packet redundancy. In router, we randomly selected one router and inserted a HT on one input port. The triggered HT duplicates any packet going through that port and connects that packet to another output port. If a deterministic routing algorithm is used, the duplicated packet will be bounced back and continue the rest of hops till reach its destination. In NI, we randomly selected one NI and allowed that NI duplicating every packet it submitted. As shown in Figure 4-14, the NI and router having a triggered HT cause the increase on the average latency. A higher packet injection rate leads to a larger average latency because of the increased traffic congestions. The latency increasing caused by the HT in router is higher than that caused by the HT in NI, when the packet injection rate is less than 0.12 packets per cycle. More precisely, as shown in Figure 4-14, the percentage of increased latency from w/o HT effect to w/ HT effect on NI is less significant than that on router. One of the reason is, the packet duplicated at NI is limited when packet injection rate is low; however, a router in the middle of NI has a higher packet throughput than a NI; thus, packet duplication on a router is more frequent than that on a NI. Consequently, the latency increased on router is severer than at on a NI. Figure 4-14 (b) also shows that, if a packet injection rate at each NI increases, the packet latency caused by HT at NI exceeds that caused by router. The variation on the percentage of increased latency from HT at router is less than that on a NI, as a nature of router reaches its maximum throughput much earlier than a NI.

```
//Pseudo code for the HT attack model of duplicating packet transmission
if(HT trigger condition is true)
    HT_payload <= on;
else
    HT_payload <= off;

//HT payload inserted during routing path calculation in a NoC router
if(HT payload == on)
Req [E, S, W, N, L]<= Route Computation in Input Port k;
    Copy ∀Req i that value is True to ∀Req j that value is false
    i,j,k∈[E,S,W,N,L],i≠k,i≠j;
    Output port i <= Data from Input FIFO k;
Output port j <= Data from Input FIFO k;
else
    Req [E, S, W, N, L]= Route Computation in Input Port k;
    Output port i <= Data from Input FIFO k;
       i,k∈[E,S,W,N,L],i≠k;

//HT payload inserted during packetization in a NoC network interface
Packetize a packet PKTm with a destination address DestAddr;
Push PKTm into NI FIFO;
NI FIFO releases PKTm to NoC fabric
if(HT payload == on)
    FIFO pointer remains same;
    NI FIFO release PKTm to NoC fabric again;
    FIFO pointer reduces one;
else
    FIFO pointer reduces one;
```

Figure 4-15 Pseudo code for HT attack model: duplicating packet transmission

We propose a pseudo code based model for this type HT. As shown in Figure 4-15, the packet duplication can be either implemented in router output port or NI FIFO. This model arises NoC designers' attention to strengthen their protocol by adding redundancy checks.

4.4.5 Payload Objective 5: Causing Buffer Overflow or Overwriting Memory

The dual-port RAM based FIFO buffer is a critical component for clock synchronization and flit queuing in NI. The FIFO full and empty signals are generated by comparing the read and write pointers to indicate the memory availability. The pointer comparison is based on circular addressing as shown inFigure4-16(a). When the write pointer is one word behind the read pointer, the FIFO full signal is asserted.

Similarly, when the write pointer is one word ahead the read pointer, the FIFO empty signal is asserted.

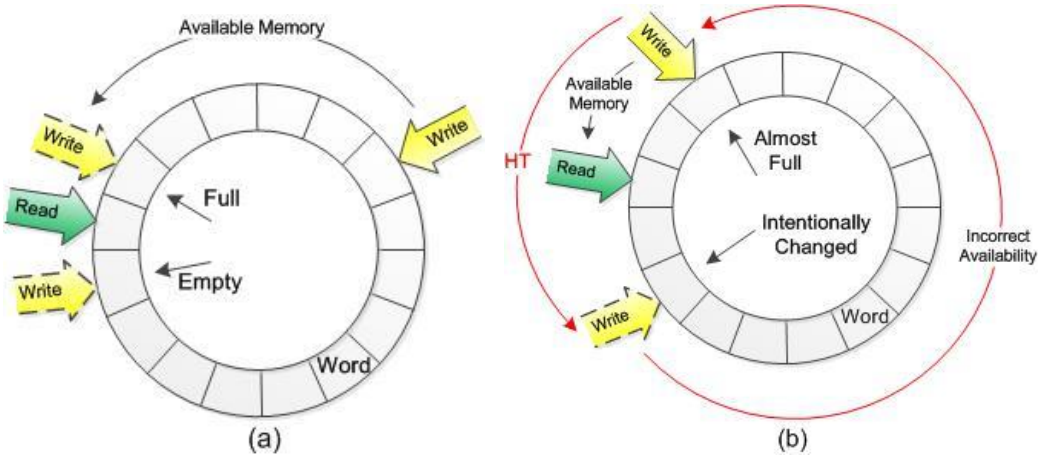


Figure 4-16 Read and write FIFO pointers in (a) normal read and write operation, and (b) HT affected situation

If a HT payload intentionally changes the pointer value, the FIFO will present incorrect memory availability signal to the external modules, as shown in Figure4-16(b). When a HT payload pushes the write pointer to the used memory location, the new incoming flits overwrite the memory and corrupt the previous packet, resulting in denial of service.

```
//Pseudo code for the HT attack model of FIFO pointers
if(HT trigger condition is true)
    HT_payload <= on;
else
    HT_payload <= off;
//HT injection location
if(HT payload == on)
case(FIFO pointer):
            Write Pointer:
        Write Pointer <= a certain address behind current address;
            Read Pointer:
        Read Pointer <= a certain address ahead of current address;
endcase
else
    Execute normal operations;
```

Figure 4-17 Pseudo code for HT attack model: manipulating FIFO

The HT attack model for malicious FIFO modification is shown in Figure 4-17.

The simple way to erase the content in FIFO is to change the read and write FIFO pointers. The changed FIFO pointer may cause an incomplete packet remain in the FIFO forever, unless the NoC has some time-out mechanism to clear the FIFO.

4.4.6 Payload Objective 5: Modifying Protocol Specific Information

Industrial on-chip interconnect protocols, such as IBM CoreConnect, ARM AXI and OCP, are widely adopted in NoC design. Once the communication protocols between IP cores are determined, the exchange packet has to explicitly include the information that the specific protocol requires. Otherwise, the received packet cannot be de-packetized correctly. The protocol information may be changed by the HTs placed in NI.

We use OCP, an open and free communication protocol [36], as an example. The *MCmd* signal defines the transfer command type of a Master IP in OCP. Changing the value of MCmd will damage a regular transaction and cause a series of unexpected errors, such as changing a write request to a read request and vice versa. *MAddr* is the transfer address signal in OCP. The adversary can make a packet access to a wrong memory address by altering this signal. Table 4-2presents a brief summary of HT attacks for basic OCP signals.

Table 4-2 Basic OCP signals

OCP Signal	Driver	Function	Possible Trojan Effects
MCmd	Master	Transfer command	Alter transfer command type
MAddr	Master	Transfer address	Alter transfer address; Hijacking

MData	Master	Write data	Corrupt Master write data
MRespAccept	Master	Master accepts response	Mute Master accepts and suspend Slave
SCmdAccept	Slave	Slave accepts transfer	Mute Slave accepts and suspend Master
SData	Slave	Read data	Corrupt Slave read data
SDataAccept	Slave	Slave accepts write data	Mute Slave accepts and suspend Master
SResp	Slave	Transfer response	Incorrect Slave responses

CHAPTER 5

HT IMPACT AND PROPOSED HT COUNTERMEASURE

This section presents the HT impact and proposed countermeasure for HTs in NI. We implemented various HTs targeted for our NI based previous analysis and HT design methodology. The HT attacks to the data transmission within the NoC were either quantitatively or visually evaluated using real digital applications. Particularly, we showed the examples fingerprint identification, image and video transmission. Finally, we proposed a state obfuscation technique for HT countermeasure in the proposed NI. The efficiency of method was evaluated, as well.

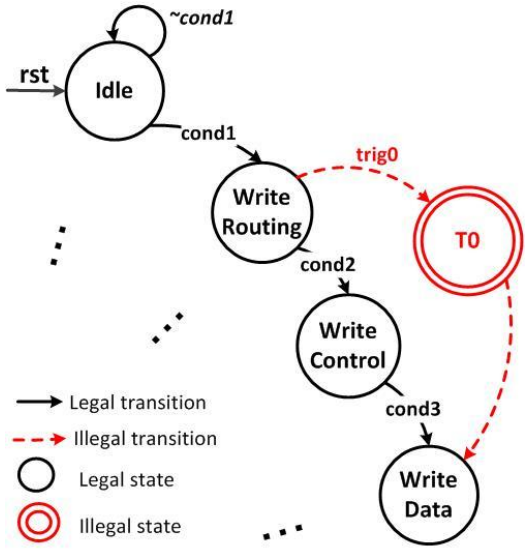
5.1 Hardware Trojan Implementation

We show three case studies to illustrate the details of HT insertion in NI, either at netlist gate level or at RTL code level. The three cases are HT insertion at FSM control, routing table and FIFO memory, respectively. We make reasonable assumptions on the attacker's objectives and explain in details how and where the HTs will be placed to conduct malicious behavior. The real implementation of various HT instances are also presented and compared in terms of area and power.

5.1.1 Case Study 1: HT Insertion in FSM Control Logic

As mentioned in Section 3, there are a great many FSM based control logic in a modular NI. Because of the FSM unit's role in NI, we expect that the FSM control unit might be a critical target point of hardware attacks. We assume that the malicious

attacker has access to the synthesized netlist of the original design and can insert HTs at the netlist level. By using EDA tools, the attacker can find out the number of registers in the FSM design and examine the FSM function through a brute-force method. HTs can therefore be placed in the FSM control unit to cause NI malfunction and degrade NoC performance.



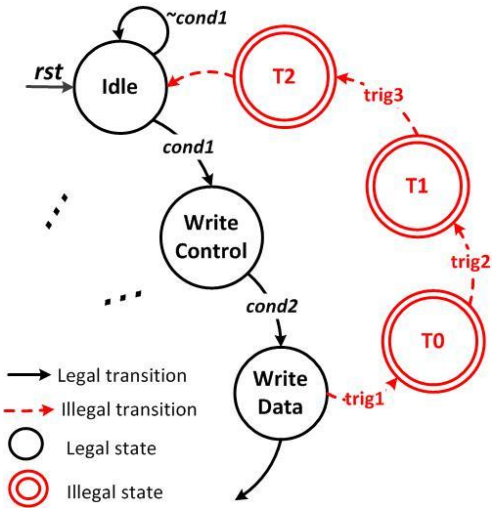

```
// Single State HT Insertion
...
// FSM next state logic
always@(*)begin
case(curr_state)
        Idle:...
        Write_Routing:begin
...
//****HT insertion***
elseif(HT_trigger)
next_state = T0;
end
T0:next_state = Write_Data;
//********************
...
default:...
end
endcase
end
//FSM output logic
always@(*)begin
case(curr_state)
        Idle:...
        Write_Routing:...
...
//****HT Payload*****
T0: wena = 1'b0;
//********************
default:...
endcase
end
```

(a) (b)

Figure 5-1 Single state HT insertion in FSM (a) conceptual view (b) RTL insertion

For an *N*-bit FSM, there are 2^N available states for the designer to construct the control logic. Typically not all the available states are utilized as legal states and the unused states create ideal places for HT insertion from attacker's perspective. The HT inserted in unused state remains dormant when in normal operation and can perform attack by disabling certain output signals. For example, Figure 5-1 (a) shows a single

state HT inserted in the NI central FSM. The HT bypasses the state of writing control information into the header flit by disabling the write enable signal to asynchronous FIFO. This leads to header flit loss to a packet and when the receiving end receives this packet, it may not be recognized as a valid packet due to the lack of control information. Figure 5-1 (b) gives the RTL insertion of HT.




```
// Multi-state HT Insertion
...
// FSM next state logic
always@(*)begin
case(curr_state)
        Idle:...
        Write_Routing:
        Write_Control:
        Write_Data:begin
if...
//****HT insertion*******
elseif(HT_trg1)
next_state = T0;
end
T0:if(HT_trg2) next_state = T1;
        T1:if(HT_trg3) next_state = T2;
        T2: next_state = Idle;
//************************
default:...
end
endcase
end
// FSM output logic
always@(*)begin
case(curr_state)
        Idle:...
        Write_Routing:...
...
//****HT Payload*****
        T0,T1,T2: wena = 1'b0;
//*******************
default:...
endcase
end
```

(a) (b)

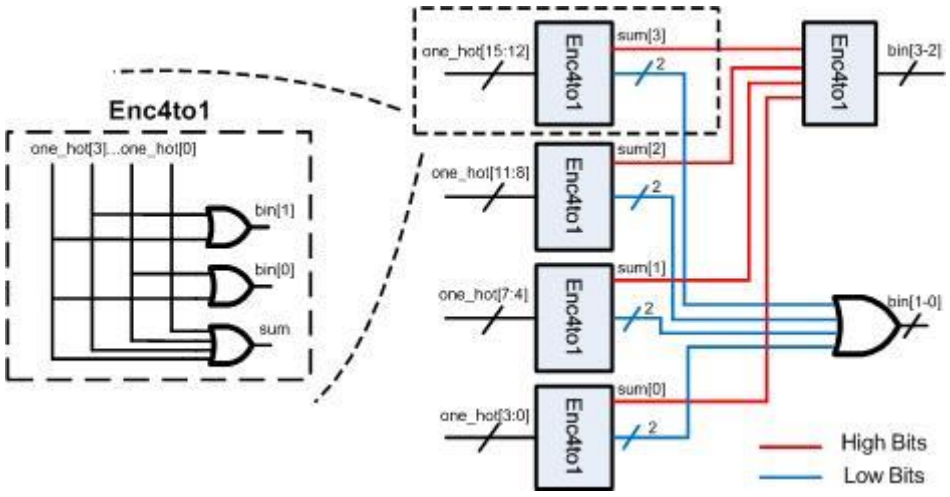
Figure 5-2 Multi-state HT insertion in FSM (a) conceptual view (b) RTL insertion

The HT in FSM can also be multi-state insertion, as shown in Figure 5-2. This HT can skip several cycle of writing data flit to the FIFO before finally being reset to idle state. In a real application of NI, the loss of data flits of a packet can also lead to serious impact. For instance, in the video stream transmission between a master CPU

and a slave MPEG decoder, the image frames of the stream may be distorted or may not be recovered due to the data loss. More details are further presented in 5.2. Similarly, Figure 5-2 (b) provides the RTL code description.

5.1.2 Case Study 2: HT Insertion in Routing Table

The routing table stores the routing information for all the IP cores within a NoC. In our design, the routing table is based on a CAM plus RAM architecture. Before fetching the routing path from the RAM, the CAM searches its contents to find a match for a certain MAddr request. The output of CAM is in one-hot code, so it has to be encoded to a binary address before getting the routing path from RAM. The one-hot to binary converter is also a place where attack can easily place HTs. Since we target for a 4×4mesh NoC, a 16-bit one-hot to 4-bit binary encoder is necessary in our NI. Figure 5-3 (a) shows the gate-level implementation in our design. The Enc4to1 sub-block realizes the 4-bit one-hot to 2-bit binary conversion and a structural implementation of 4 Enc4to1 sub-blocks can realize the 16-bit to 4-bit conversion. Figure 5-3 (b) shows the RTL insertion of HT. If the binary read address to RAM is hacked, an incorrect routing path will be assigned to a certain packet, and will eventually cause livelock or deadlock in NoC.



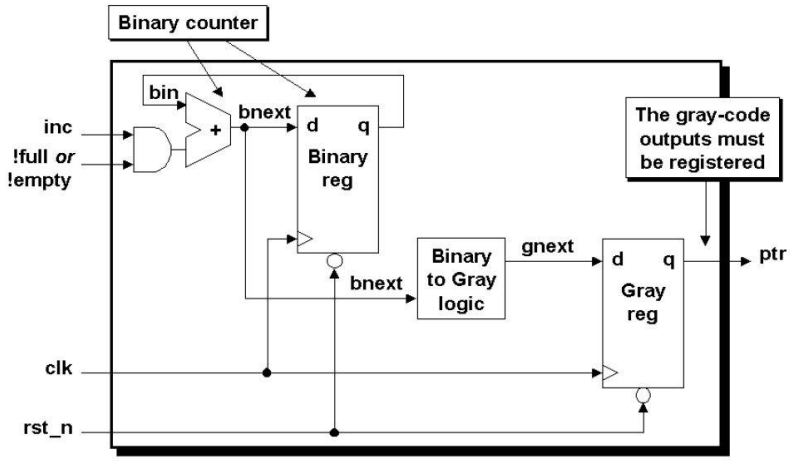
(a)

```
// HT at 16/4 One-hot Binary Encoder
// 4-bit one-hot to 2-bit binary Encoder
module Enc4to2(one_hot, bin, sum,HT_trigger);
input[3:0] one_hot;
output[1:0] bin;
output sum;
wire[1:0] bin;
wire   sum;// sum is true if any input is true
//**********HT Insertion*****************
inputHT_trigger;
assign bin[1]=(one hot[3]| one hot[2])^HT_trigger;
assign bin[0]=(one_hot[3]| one_hot[1])^HT_trigger;
//**************************************
assign sum =|one hot;
endmodule
//factored 16-bit to 4-bit encoder
module Enc16to4(one hot, bin,HT_trigger);
input[15:0] one hot;
output[3:0]  bin;
wire[3:0] bin;
wire[7:0]temp;// intermediate result of first stage
wire[3:0] d;// if any set in group of four
//two OR gates combine output of LSB
//**********HT Insertion*****************
inputHT_trigger;
assign bin[1]=(temp[1]|temp[3]|temp[5]|temp[7])^HT_trigger;
assign bin[0]=(temp[0]|temp[2]|temp[4]|temp[6])^HT_trigger;
//**************************************
//four LSB encoders each include 4-bits of the input
  Enc4to2 e0(one_hot[3:0],temp[1:0],d[0]);
  Enc4to2 e1(one hot[7:4],temp[3:2],d[1]);
  Enc4to2 e2(one hot[11:8],temp[5:4],d[2]);
  Enc4to2 e3(one_hot[15:12],temp[7:6],d[3]);
//MSB encoder takes summaries and gives MSB
  Enc4to2 e4(d[3:0], bin[3:2]);
endmodule
```

(b)

Figure 5-3 HT insertion in the one-hot to binary encoder of routing table (a) gate-level implementation of one-hot binary encoder (b) HT insertion at RTL

5.1.3 Case Study 3: HT Insertion in FIFO Read/Write Pointer



(a)

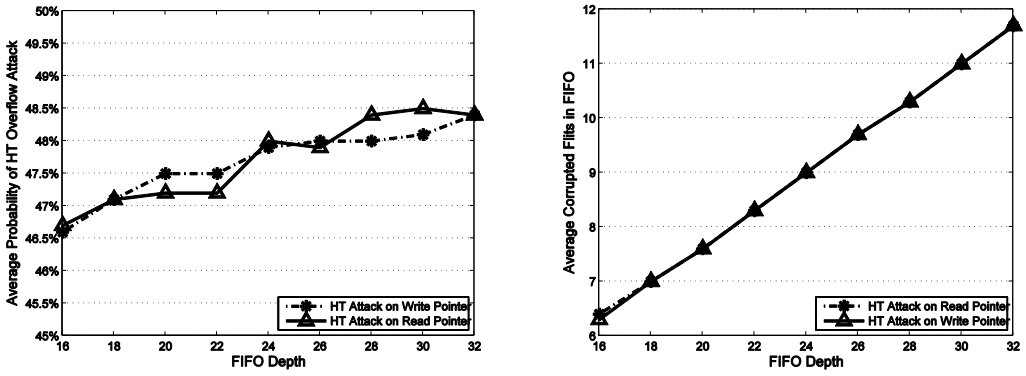
```
// Gray code read pointer generation
module rptr_empty(...);
...
output rempty;// FIFO read empty
parameter ADDRSIZE =4;
wire[ADDRSIZE-1:0] rgnext, rbnext;
...
//*************HT Insertion*****************************
input[ADDRSIZE-1:0]HT_trigger;
//Insertion at Binary counter
assign rbnext =(!rempty ? rbin + rinc : rbin)^HT_trigger;
//Insertion at Gray code conversion
assign rgnext =((rbnext>>1)^ rbnext)^HT_trigger;
//*******************************************************
...
endmodule
```

(b)

Figure 5-4 HT insertion in FIFO pointer (a) gate-level implementation of NI FIFO pointer (b) RTL insertion of HT

The HT can also be placed in the FIFO memory itself, for example, the FIFO pointer and its related output signals. The FIFO read and write pointers are indicator of the memory availability and thus controlling the full and empty signals to the external. Figure 5-4 (a) presents the general view of binary and Gray code combined pointer used in our design. Binary counter is designed for pointer incrementing and addressing to the FIFO RAM. When the RAM is not full (empty), the binary write

(read) counter increments as the *inc* signal asserts. The Gray code counter is designed for fast pointer comparison to assert or de-assert the FIFO full and empty signals. Figure 5-4 (b) presents the HT attack to the write pointer at RTL. If the binary or Gray code counters are flipped when the HT trigger asserts, they will generate incorrect full or empty signals to external modules, or incorrect binary address to the internal RAM. Both cases will lead to FIFO overflow and data corruption.



(a) (b)

Figure 5-5 HT attack in FIFOs with different depth (a) the probability causing overflow (b) the average number of corrupted data flit

We justify that the HT attacks to FIFO pointer may possibly cause memory overflow or overwriting by giving incorrect availability information to the external. However, this situation only happens when writer pointer is changed to a certain address behind current address or read pointer is changed to an address ahead of current address. The results of HT attack may be slightly different over different FIFO depth due to its different addressing. If we assume the HT randomly flips the FIFO pointers, Figure 5-5 (a) shows the probability that memory overflow happens. The probability is slightly increasing but still in the range of 46% to 48.5%. Figure 5-5 (b)

shows the average number of corrupted flits if such attack happens. The number of corrupted flits is increasing almost linearly with the FIFO depth, meaning the impact of attack will become more serious in a larger FIFO.

5.1.4 HT Hardware Overhead

Table 5-1and 5-2give the implementation details of several practical HTs discuss previously. HT trigger circuits can be combinational, asynchronous counter based, hybrid and FSM based, which are shown in Table 5-1; HT payloads can be placed at global clock, routing table, FIFO pointer, FSM and header flit links, which are shown in Table 5-2.

Table 5-1Implementation result summary of HT trigger circuits

Trig. Type	Circuit Structure	Comb. Gate	NO. of DFFs
Comb. Trigger	1 XOR	1	0
Async. Counter	XOR+4-bit counter	14	4
Hybrid	XOR+2 4-bit counters	27	8
FSM (a)	4-state FSM	7	2
FSM (b)	4-state FSM	6	2
FSM (c)	4-state FSM	8	2
FSM (d)	4-state FSM	7	2

Table 5-2 Implementation result summary of HT payload circuits

Payload	Location	Comb. Gate	Effect
Payload1	global clock	1	Latching clock source
Payload2	routing table	2	Causing incorrect routing path
Payload3	FIFO pointer	4	Causing FIFO overflow
Payload1	FSM logic	NA	Causing header/payload flit loss
Payload5	header flit links	2	Damaging flit type

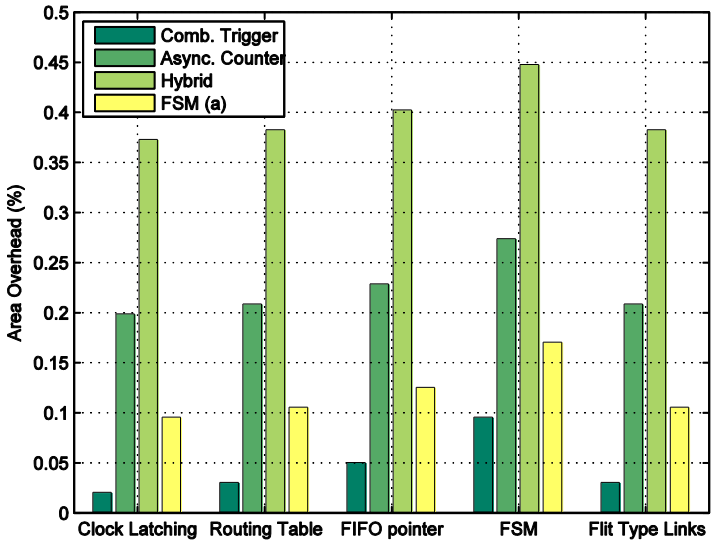


Figure 5-6 Comparison of different HTs in terms of area

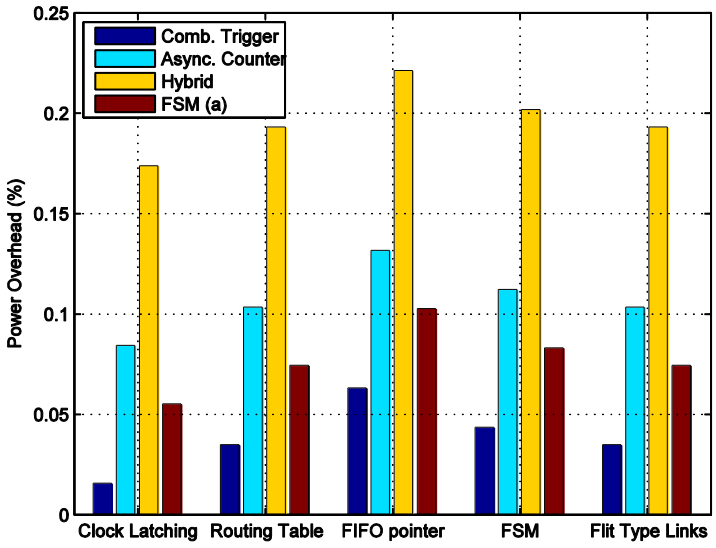


Figure 5-7 Comparison of different HTs in terms of power

The synthesized results in IBM 0.18μm technology are also provided in Figure 5-6 and Figure 5-7 in terms of area and power, respectively. HT instances present trivial overhead in all cases, less than 0.45% in area and less than 0.25% in power. Among them, combinational triggered HTs present minimum result, hybrid triggered HTs present maximum overhead.

5.2 HT Impact From Application Perspectives

Though the HT insertion in NI may have various types of instances, the main effect of HT is to cause data or header flit loss through different ways. In this section, we provide two digital application examples visually evaluate the impact of flit loss caused by HTs. More specifically, we show the examples of the video stream transmission and fingerprint identification.

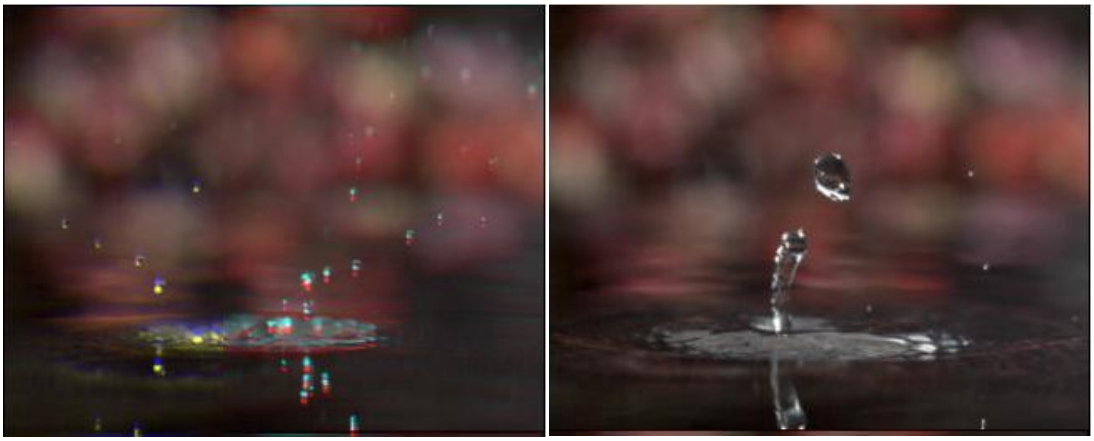
5.2.1 Flit Loss in Image & Video Transmission

Lost flits in the data transmission of NI will present significant impact to the image and video applications. Assuming that a NI is the bridge between a MPEG decoder and a graphical processor, we conduct experiments to simulate a series of image frames or video fragments are transmitting via the NI, which are shown as in Figure 5-8 (a) to (c). Each image is in a 326×260 RGB pixel format and finally converted to 63570 32-bit data flits when being transmitted. The lost or corrupted flits will distort or shift the images at display. Figure 5-9 shows the situation when only single data flit is lost. The distortion is noticeable in the first image frame, but the second image is still in great quality with negligible shifting. Figure 5-10 and 5-11 presents the cases of six adjacent flits lost and six random flits lost, respectively. If the flits are adjacent, the first image are seriously distorted but the second image are only shifted with the information of next image frame, as shown in Figure 5-10; if the lost flits are randomly chosen, both of the images are distorted and shifted, which are shown as in Figure 5-11. In all the cases above, the last image frame cannot be

recovered since there is no enough flit data to recover a complete image frame.

(a) (b) (c)

Figure 5-8 A series of image frames for video stream



(a) (b)

Figure 5-9 Recovered image with single data flit loss

(a) (b)

Figure 5-10 Recovered image with multiple adjacent data flits loss

(a) (b)

Figure 5-11 Recovered image with multiple random data flits loss

5.2.2 Flit Loss in Fingerprint Scanning & Identification

Flit loss may also cause similar impact to identification or verification application. Assuming the NI is transmitting data from a fingerprint scanner to a central processor, the corrupted flits will make the identification more difficult. The original fingerprint is shown in Figure 5-12 (a). Similarly, the fingerprint image is first converted to a long sequence of binary data flits, and we simulate the case that image is corrupted during transmission by flipping the binary information. We explore the cases that the fingerprint image is corrupted with 0.1%, 0.5% and 1% data flits when transmitted, respectively. The recovered fingerprints at the receive end are shown in Figure 5-12 (b) to (d), correspondingly. The corrupted data are negligible in Figure 5-12 (b), become noticeable in Figure 5-12 (c) and continue to be more serious in Figure 5-12 (d). We expect the image quality will further degrade with more data flits are corrupted.

(a) (b)

(c)(d)

Figure 5-12 HT impact to fingerprint identification (a) original fingerprint (b) fingerprint with 0.1% flit data corrupted (c) fingerprint with 0.5% flit data corrupted (d) fingerprint with 1% flit data corrupted

5.3 Proposed HT Countermeasure

We discussed the HT insertion and HT impact to data transmission in previous sections of this chapter. In this section, we explored the countermeasure technique for HT detection at runtime. More specifically, we propose to exploit state obfuscation to facilitate hardware Trojan (HT) detection in the network interface (NI) of Network-on-Chips (NoCs). Secure memory access in NoCs has been studied in

previous work. Unfortunately, in addition to memory blocks, HT payloads might be inserted at other places in NoCs to cause NoC malfunctions and allow unauthorized accesses. In the first phase of our method, we add key bits to the finite state machine for the NI control unit and create dummy states to increase the difficulty for the HT attacker to perform meaningful attacks. In the second phase, we examine the illegal states and illegal state transitions induced by a wrong key to detect the occurrence of HTs. Similar concepts, e.g. logic encryption and obfuscation [50, 51], were mainly used for IP authentication, rather than HT detection. In [50], the obfuscation modes are decoupled from the normal operation modes and need be executed before the real function starts. In contrast, our method tightly integrates the obfuscation states with normal states. Moreover, we exploit the obfuscation states to detect HTs at runtime, with minor area and power overhead. The HT detection efficiency of our method is also evaluated.

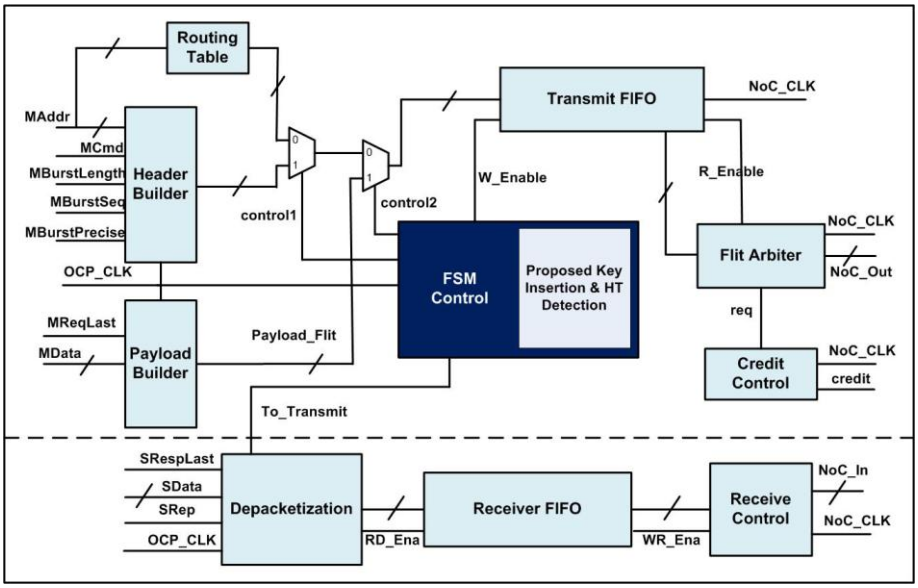


Figure 5-13 Initiator NI with embedded HT detection

5.3.1 Potential Security Threats to FSM

As we discussed before, the FSM control unit in NI contains the main control logic of the transmitter. The FSM is synchronous to the OCP clock, and is responsible for reordering and writing the flits into transmit FIFO after packetization. Because of the FSM unit's role in NI, we argue that the FSM control unit might be a critical target point of hardware attacks. In our method, we assume that HTs are placed in the FSM control unit to cause NI malfunction and degrade NoC performance. Our HT countermeasure method is thus embedded in the FSM control unit, as shown in dark shadow area in Figure 5-13. The details of state transition are presented again in Figure 5-14.

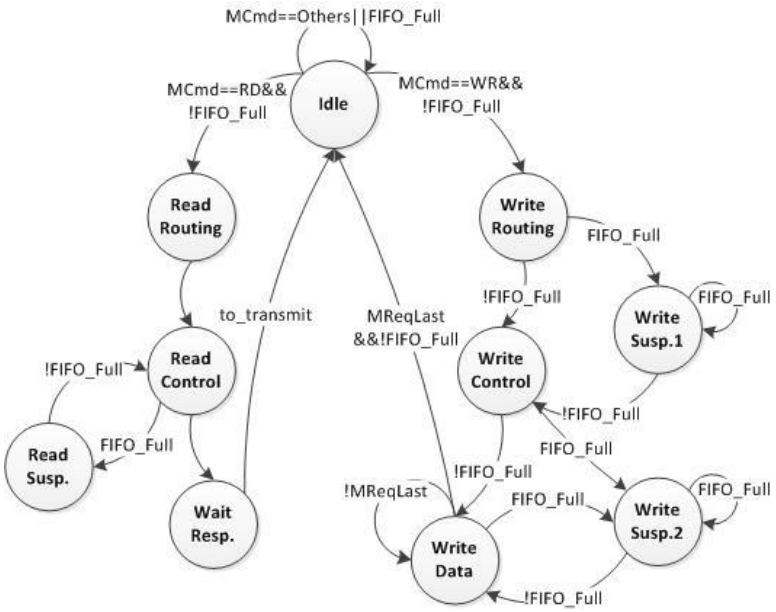


Figure 5-14 FSM for Initiator NI

5.3.2 Proposed State-Obfuscation HT Countermeasure and Detection Method

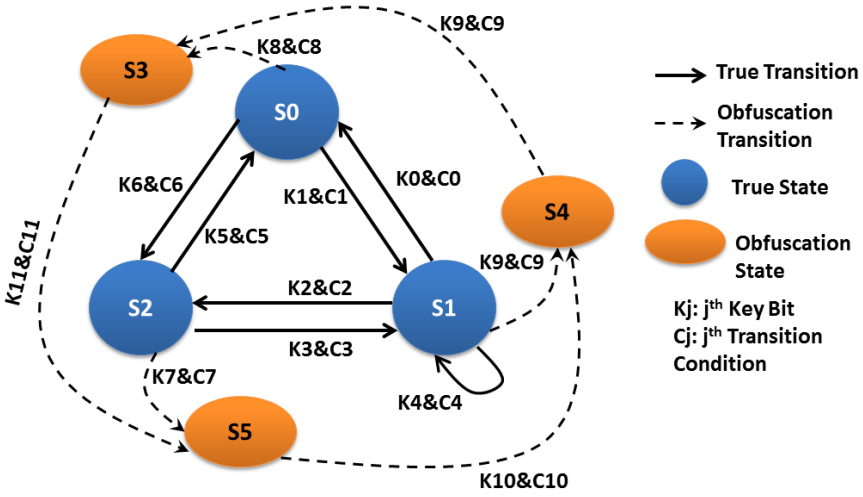


Figure 5-15 New FSM diagram after using the proposed state-obfuscation method

Our method is composed of two phases: (1) insert key bits to the FSM state transition conditions, and (2) detect illegal states and illegal state transitions. The key idea for the first phase is shown in Figure 5-15. One key bit is added to the condition of each state transition. Without knowing the key bits, the attacker cannot precisely control the effect of the HT insertion. If randomly modifying the FSM logic, the

attacker may lead the FSM turn into a default or illegal state. To confuse the attacker, we add obfuscation states in the FSM and connect each true state to a dummy state. If hardware cost is restricted, multiple true states can share one dummy state. Moreover, all the dummy states are connected, so that the FSM cannot jump back to a legal state after the FSM enters one of the illegal and dummy states. This arrangement facilitates to increase the HT detection rate. To further confuse the attacker, we can also put a key bit and a dummy transition condition between two dummy states. Examples of two detectable scenarios are shown in Figure 5-16.

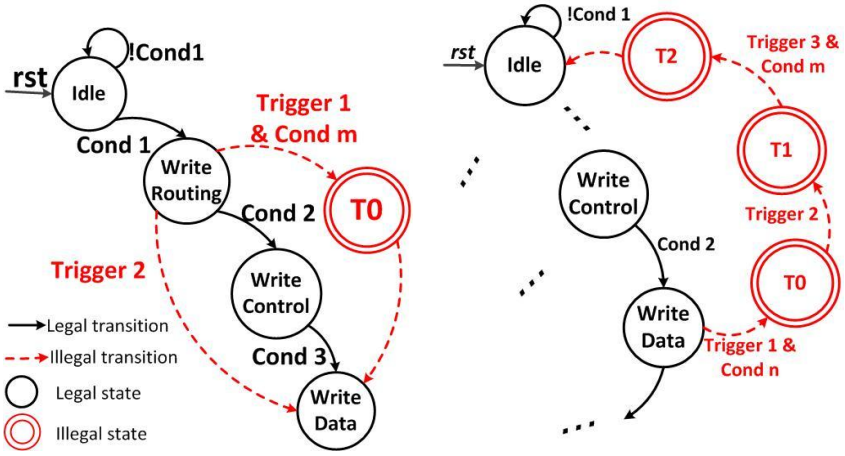


Figure 5-16 Examples of detectable HT insertion cases in FSM (a) Single and (b) multiple illegal states and illegal state transitions

The second phase is to examine whether two consecutive FSM states are defined in the FSM. To save the previous FSM state, we double the registers for the FSM states. The abnormal state transitions together with undefined FSM states are detected in the illegal state and illegal state transition detector unit, as shown in Figure 5-17. Dummy states introduced in the first phase are a portion of the illegal states. As the key sequence and the location of each key bit are unknown to the attacker, the effect

of HTs is likely to be one of our detectable illegal cases in the detector.

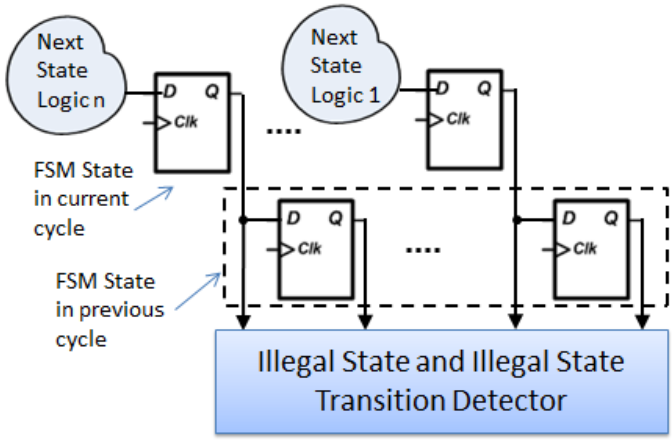


Figure 5-17 Proposed HT detector that checks illegal states and illegal state transitions

5.3.3 Simulation Results

We applied the proposed state-obfuscation based HT detection method to the NIs. We implemented the NI a TSMC 65nm technology. Post-synthesize simulation was performed in Cadence NC-Verilog. To validate the efficiency of our method, we examined the HT detection rate for the situation that the HT payloads were placed in flip-flops and logic gates of the NI FSM control unit. For single- and double-HT insertion cases, we examined all possible cases. To save simulation time, we use Monte Carlo random simulation method to examine the impact of three, four and five HTs insertion on logic gates. 2,400,000 simulation cases were executed for each data point.

- *HT Detection Rate*

We believe that security check on NIs is vital to ensure the security and availability of the entire network and the system employing the NoC. The FSM

control unit is the most critical component in the NI, as the flit packetization and OCP protocol interpretation are mainly controlled by the FSM control unit. Malfunction on NI causes packet loss, unauthorized IP access and NoC performance degradation. In the following experiments, we examined our HT detection rate in the FSM control unit. Without a high HT detection rate, the risk of NI functionality being modified by malicious hackers is correspondingly high.

In our NI FSM, we utilized five bits to represent different states. We assumed that the most effective way to hack the FSM is directly inverting the content stored in the FSM registers. Therefore, we examined the HT detection rate under the occurrence of one to five HTs in the FSM unit. As *fifo_full*, *MCmd*, *MReqLast* and *to_transmit* are the control signals (shown in Figure 5-14) to trigger the FSM transitions, we also varied those control signals in our experiments. We define the HT detection rate as the ratio of HT detected cases over the total evaluation cases. As shown in Figure 5-18, the proposed HT detection method achieves over 97.5% HT detection rate in various test cases. If the number of HT payloads increases to four and five, our method obtains 100% HT detection rate. The HT detection rates for one HT and two HTs are slightly lower than other cases; this is because Hamming distance between two FSM states is one. We expect that the HT detection rate can be further improved by carefully assigning binary value for each state.

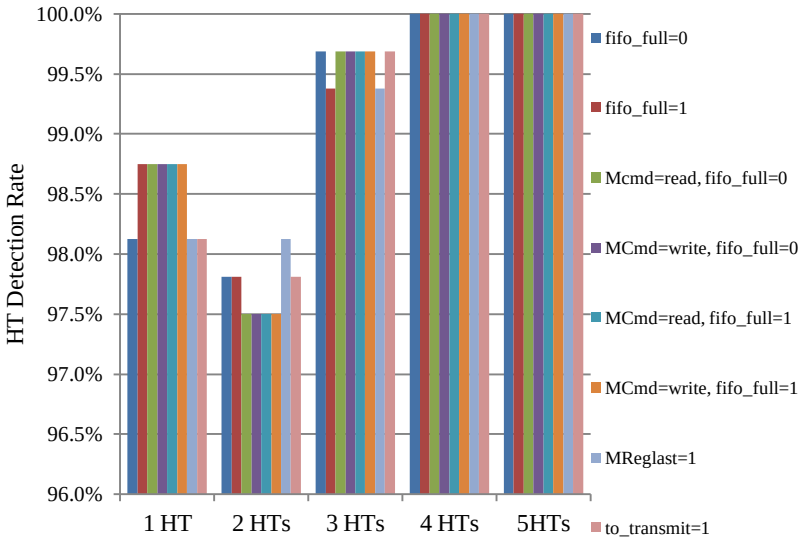


Figure 5-18 HT detection rate for HT attacks in the FSM state registers

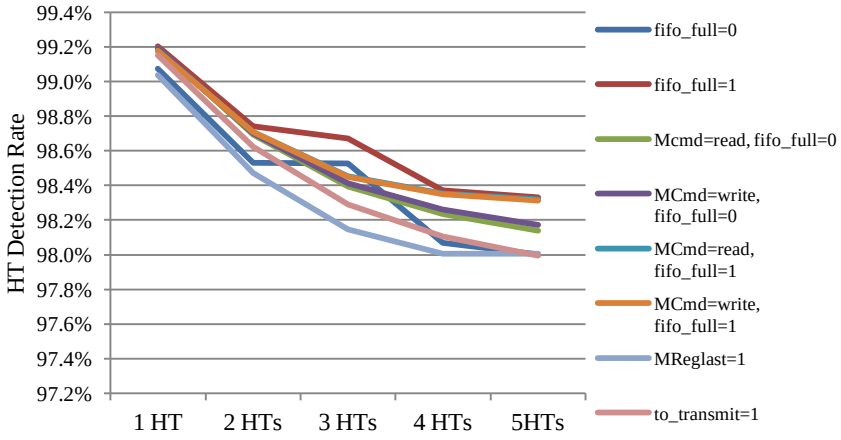


Figure 5-19 HT detection rate for HT attacks in the FSM logic gates

Certainly, HT payload can also be placed in any logic gates in the FSM control unit. In fact, it is easier to hide the HT insertion in logic gates than to hide in the FSM registers. However, the former one is less effective than the latter one. One of the reasons is, the effect of the HT payload in logic gate maybe diminish due to logical masking. As shown in Figure 5-19, the HTs inserted in logic gates can be detected

with a probability of over 98%. Generally, the HT detection rate for less HT payloads is higher than that for more HT payloads, as the overall effect of less HTs are more likely to be filtered by logic inherent masking effects.

- *Reduction on HT Attack Success Rate*

We define the HT attack success rate as the probability of a HT successfully changing the FSM from one legal state to another legal state. As we discussed in Section 3, the application of key and dummy states in the FSM increases the number of possible states and state transitions. Without knowing the key, the HT inserted is very likely to lead the FSM enter the illegal states or illegal state transition. By using the proposed method, we can reduce the attack success rate of HTs inserted by attackers, even when the FSM state transition diagram is leaked.

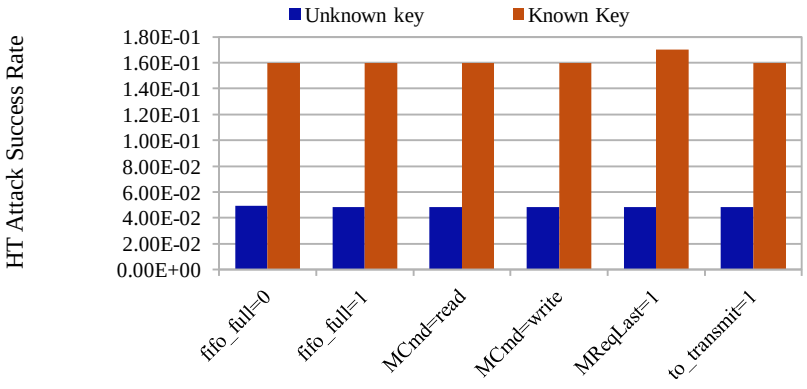


Figure 5-20 Impact of key knowledge on HT attack success rate

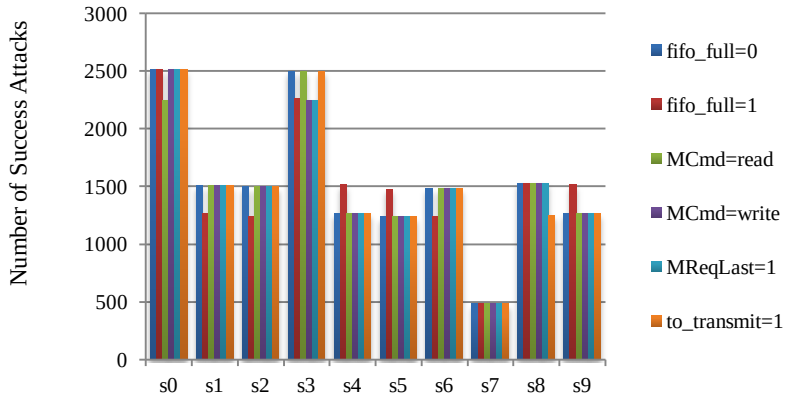


Figure 5-21 Impact of different FSM states on HT successfully attacks

As shown in Figure 5-20, without the correct key on the FSM state transition diagram, the probability of a HT successfully modifying the FSM state is 71.5% less than that of knowing the exact key value. We further examined the impact of different FSM states and control signals on the HT attack success rate. As shown in Figure 5-21, the most frequently used states (e.g. s0 and s3) are easier to be changed by HTs than other states. When the control signal associated with the particular state is asserted, the impact of HTs is more significant than the control signal is not enabled.

- *Area and Power Comparison*

The proposed method achieves a high HT detection rate and reduces the probability of a HT attack successfully modifying the FSM. As the critical path for NI w/wo our method is less than 1ns, we set the clock frequency of the NIs to 500MHz. As shown in Table 5-3, our method has 3.2% area overhead and consumes 1.7% more total power, compared to the baseline NI design.

Table 5-3 Area and Power of the NIs w/wo Proposed Method

Design for Comparison	Area (μm^2)	Power (mW)
Baseline NI (no key insertion and HT detection)	19133.2 (100%)	6.491 (100%)
NI with proposed state obfuscation and HT detection	19747.7 (103.2%)	6.605 (101.7%)

CHAPTER 6

CONCLUSIONS

Network-on-Chip (NoC) is emerging as a prevalent on-chip communication infrastructure. The flexibility and scalability of NoC make it feasible to migrate to the era of many heterogeneous IP cores on a single die. As the scale and complexity of a NoC system increase, the security of on-chip communication is expected to be another concern. Although the globalization of current IC industry helps to reduce design cost and shorten the time to market, it also makes the fabricated chips vulnerable to hardware tampering and Trojan insertion. In this thesis, a highly modular network interface is designed and implemented for OCP compatible NoC systems. The hardware security aspects of NI are analyzed and addressed.

For the future work in this topic, the HT implementation in a system level emulation platform using FPGA or ASIC technology is highly expected. Also, more advanced and efficient HT countermeasure techniques need to be explored as well. The key contributions of this thesis are summarized as follows.

This thesis starts with basic introduction of NoC development, hardware security issues in IC chips. In Chapter 2, we review the basics of NoC design problems. Based on the design problems analyzed, we proposed our baseline NI design in Chapter 3. The proposed NI was implemented with an IBM 0.18μm CMOS technology and compared with exiting designs reported in literature. The synthesis results in terms of

power, timing and area reported. The hardware cost of our NI design is comparable performance with previous work.

In Chapter 4, we analyzed the network interface circuit at the system level and presented comprehensive and meaningful HT attack models from attackers' perspectives. The impact of multiple practical HT triggers and payloads on NI performance and NoC applications are evaluated with simulations. Chapter 5 presents the detailed implementation results of HTs. The corresponding effect of HT insertion was also assessed with real digital application examples. Finally, a state obfuscation technique is proposed for HT countermeasure and detection. Our method obtains a relatively high detection rate of over 98% for all test cases and reduces the HT attack success rate by 71.5%, at the cost of 3.2% area increase and 1.7% more power consumption in the TSMC 65nm CMOS technology.

REFERENCES

[1] J. D. Owens, W. J. Dally, R. Ho, D. Jayasimha, S. W. Keckler and L. S. Peh, “Research Challenges for On-Chip Interconnection Networks,” *IEEE Micro,* vol. 27, no. 5, pp. 96-108, 2007.

[2] ARM Ltd., “AMBA Specifications,” 2013. [Online]. Available: http://www.arm.com/products/system-ip/amba/amba-open-specifications.php.

[3] IBM MicroelectronicsIBM, “CoreConnect Bus Architecture,” January 2006. [Online]. Available: https://www-01.ibm.com/chips/techlib/techlib.nsf/productfamilies/CoreConnect_Bus_Architecture.

[4] OpenCores, “SoC Interconnection: Wishbone,” [Online]. Available: http://opencores.org/opencores,wishbone.

[5] Arteris, Inc., “A comparison of Network-on-Chip and Busses,” 2005. [Online]. Available: http://www.design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html.

[6] A. Agarwal, C. Iskander and R. Shankar, “Survey of Network on Chip (NoC) Architecture & Contributions,” *Journal of Engineering, Computing and Architecture,* vol. 3, no. 1, 2009.

[7] L. Fiorin, C. Silvano and M. Sami, "Security Aspects in Networks-on-Chips: Overview and Proposals for Secure Implementations," in *Proc. of 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, Aug. 2007.

[8] R. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," in *Proc. of IEEE International High Level Design Validation and Test Workshop,* Nov. 2009.

[9] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang and Y. Zhou, “Designing and implementing malicious hardware,” in *Proc. of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats,* 2008.

[10] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Proc. of International Conference on Cryptographic Hardware and Embedded Systems*, Sept. 2012.

[11] S. Adee, “The Hunt For The Kill Switch,” *IEEE Spectrum* , vol. 45, no. 5, pp. 34-39, May, 2008.

[12] S. Saponara, T. Bacchillone, E. Petri, L. Fanucci, R. Locatelli and M. Coppola, “Design of a NoC Interface Macrocell with Hardware Support of Advance Networking Functionalities,” *IEEE Trans. on Computers,* vol. 63, pp. 609-621, March 2014.

[13] S. Evain and J.-P. Diguet, "From NoC security analysis to design solutions," in *IEEE Workshop on Signal Processing Systems Design and Implementation*, Nov. 2005.

[14] J.-P. Diguet, S. Evain, R. Vaslin, G. Gogniat and E. Juin, "NOC-centric Security of Reconfigurable SoC," in *Proc. of First International Symposium on Networks-on-Chip (NOCS)*, May 2007.

[15] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano and C. Silvano, "Secure Memory Accesses on Networks-on-Chip," *IEEE Trans. on Computers*, vol. 57, no. 9, pp. 1216-1229, Sept. 2008.

[16] C. Gebotys and R. Gebotys, "A framework for security on NoC technologies," in *Proc. of IEEE Computer Society Annual Symposium on VLSI*, Feb. 2003.

[17] C. Gebotys and Y. Zhang, "Security wrappers and power analysis for SoC technology," in *Proc. of First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, Oct. 2003.

[18] K. Sajeesh and H. Kapoor, "An Authenticated Encryption Based Security Framework for NoC Architectures," in *Proc. of 2011 International Symposium on Electronic System Design (ISED)*, Dec. 2011.

[19] M. LeMay and C. A. Gunter, "Network-on-Chip Firewall: Countering Defective," 2014. [Online]. Available: http://arxiv.org/abs/1404.3465.

[20] P. Kocher, R. Lee, G. McGraw, A. Raghunathan and S. Ravi, "Security as a new dimension in embedded system design," in *Proc. of 41st Design Automation Conference*, July 2004.

[21] E. Cota, A. d. M. Amory and M. S. Lubaszewski, "Chapter2: NoC Basics," in *Reliability, Availability and Serviceability of Networks-on-Chip*, New York City, NY, Springer, 2012, pp. 11-24.

[22] E. Salminen, A. Kulmala and T. D. Hamalainen, "Survey of Network-on-Chip Proposals," OCP-IP, March 2008. [Online]. Available: http://ns2.ocpip-server.com/uploads/documents/OCP-IP_Survey_of_NoC_Proposals_White_Paper_April_2008.pdf.

[23] G. De Micheli and L. Benini, Network-on-Chips: Technology and Tools, San Francisco, CA: Morgan Kaufmann, 2006.

[24] OCP-IP, "OCP Specifications 3.0," [Online]. Available: http://www.ocpip.org/uploads/dynamic_areas/Xu4qydXgbYWof7Ihz3Uh/947/Open%20Core%20Protocol%20Specification%203.0.pdf.

[25] W. J. Dally and B. Towles, Principles and Practices of Interconnection Networks, San Francisco, CA: Morgan Kaufmann, 2004.

[26] T. Bjerregaard, S. Mahadevan, R. Olsen and J. Sparsoe, "An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip," in *Proc. of International Symposium on System on Chip*, Nov. 2005.

[27] S. E. Lee, J. H. Bahn, Y. S. Yang and N. Bagherzadeh, "A Generic Network Interface Architecture for a Networked Processor Arrary (NePA)," in *Proc. 21st International Conference on Architecture of Computing Systems (ARCS)*, Dresden, Germany, Feb. 2008.

[28] M. Mitic, M. Stojcev and Z. Stamenkovic, “An Overview of SoC Buses,” in *The Computer Engineering Handbook: Digital Systems and Applications*, Boca Raton, Florida, CRC Press, 2007.

[29] C. Grecu, A. Ivanov, R. Pande, A. Jantsch, E. Salminen, U. Ogras and R. Marculescu, "Towards Open Network-on-Chip Benchmarks," in *Proc. of First International Symposium on Networks-on-Chip*, May 2007.

[30] T. Bjerregaard and J. Sparso, "Packetizing OCP Transactions in the MANGO Network-on-Chip," in *Proc. of 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, 2006.

[31] B. Attia, A. Zitouni, W. Chouchenne, K. Torki and R. Tourki, “A Modular Network Interface Design and Synthesis Outlook,” *International Journal of Computer Science (IJCSI)*, vol. 9, no. 3, pp. 470-482, May 2012.

[32] L. Fiorin and M. Sami, "Fault-Tolerant Network Interfaces for Networks-on-Chip," *IEEE Trans. on Dependable and Secure Computing*, vol. 11, no. 1, pp. 16-29, 2014.

[33] L. Fiorin, L. Micconi and M. Sami, "Design of Fault Tolerant Network Interfaces for NoCs," in *Proc. of 14th Euromicro Conference on Digital System Design (DSD)*, August 2011.

[34] Altera Corporation, "Implementing High-Speed Search Applications with Altera CAM," 2001. [Online]. Available: www.altera.co.jp/literature/an/an119.pdf.

[35] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: a tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712-727, 2006.

[36] M. Peng and S. Azgomi, "Content-Addressable memory (CAM) and its network applications," Altera International Ltd., 2000.

[37] I. Loi, F. Angiolini and L. Benini, "Synthesis of low-overhead configurable source routing tables for network interfaces," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, April, 2009.

[38] M. Akhbarizadeh, M. Nourani, D. Vijayasarathi and P. Balsara, "PCAM: a ternary CAM optimized for longest prefix matching tasks," in *Proc. of International Conference on Computer Design: VLSI in Computers and Processors (ICCD)*, Oct. 2004.

[39] C. E. Cummings and P. Alfke, “Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons,” Sunburst Design, Inc., 2002. [Online]. Available: http://www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO2.pdf.

[40] C. E. Cummings, “Simulation and Synthesis Techniques for Asynchronous FIFO Design,” Sunburst Design, Inc., 2002. [Online]. Available: www.sunburst-design.com/papers/CummingsSNUG2002SJ_FIFO1.pdf.

[41] T. T. Ye, L. Benini and G. De Micheli, "Packetized On-chip Interconnect Communication Analysis for MPSoC," in *Proc. of the Design Automation and Test in Europe Conference and Exhibition (DATE)*, 2003.

[42] Y.-L. Lai, S.-W. Yang, M.-H. Sheu, Y.-T. Hwang, H.-Y. Tang and P.-Z. Huang, "A High-Speed Network Interface Design for Packet-Based NoC," in *Proc. of International Conference on Communications, Circuits and Systems Proceedings*, June 2006.

[43] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi and G. De Micheli, "×pipes Lite: a synthesis oriented design library for networks on chips," in *Proc. of Design, Automation and Test in Europe*, March 2005.

[44] A. Radulescu, J. Dielissen, K. Goossens, E. Rijpkema and P. Wielage, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration," in *Proc. of Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Feb. 2004.

[45] D. Matos, M. Costa, L. Carro and A. Susin, "Network interface to synchronize multiple packets on NoC-based Systems-on-Chip," in *Proc. of 18th IEEE/IFIP VLSI System on Chip Conference (VLSI-SoC)*, Sept. 2010.

[46] M. Tehranipoor and C. Wang, Introduction to Hardware Security and Trust, New York, NY: Springer, 2012.

[47] X. Wang, S. Narasimhan, A. Krishna, T. Mal-Sarkar and S. Bhunia, "Sequential hardware Trojan: Side-channel aware design and placement," in *Proc. of IEEE 29th International Conference on Computer Design (ICCD)*, Oct. 2011.

[48] J. Zhang and Q. Xu, "On hardware Trojan design and implementation at register-transfer level," in *Proc. of IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, June 2013.

[49] Y. Jin, N. Kupp and Y. Makris, "Experiences in Hardware Trojan design and implementation," in *Proc. of IEEE International Workshop on Hardware-Oriented Security and Trust*, July 2009.

[50] R. Chakraborty and S. Bhunia, "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493-1502, Oct. 2009.

[51] J. Rajendran, Y. Pino, O. Sinanoglu and R. Karri, "Logic encryption: A fault analysis perspective," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2012.

[52] R. Karri, J. Rajendran, K. Rosenfeld and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *IEEE Computer* , vol. 43, no. 10, pp. 39-46, Oct. 2010.

[53] L. Benini and G. De Micheli, "Network on Chips: A New SoC Paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70-78, 2002.

[54] P. P. Pande, G. De Micheli, C. Grecu, A. Ivanov and R. Saleh, "Design, synthesis, and test of networks on chips," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 404-413, 2005.

[55] Altera Corporation, "Avalon Interface Specifications," 2014. [Online]. Available: www.altera.com/literature/manual/mnl_avalon_spec.pdf.

[56] K. Goossens, J. Dielissen and A. Radulescu, “Aethereal Network on Chip: Concepts, Architectures, and Implementations,” *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 414-421, 2005.

[57] T. Bjerregaard and S. Mahadevan, “A Survey of Research and Practices of Network-on-Chip,” *Journal of ACM Computing Surveys (CSUR)*, vol. 38, no. 1, pp. 1-51, 2006.

[58] S. Butler, “Managing IP quality in the SoC era requires a purpose-built DM approach,” Methodics LLC, Sept. 2011. [Online]. Available: http://www.eetimes.com/author.asp?section_id=36&doc_id=1266011.

[59] S. Malviya and A. Jaiswal, “Five Port Router for Network on Chip,” in *Proc. of ARRL and TAPR Digital Communications Conference*, 2010.

[60] V. Rantala, T. Lehtonen and J. Plosila, “Network on Chip Routing Algorithms,” Turku Center for Computer Science, August, 2006.

[61] M. Beaumont, B. Hopkins and T. Newby, "Hardware Trojans – Prevention, Detection, Countermeasures (A Literature Review)," DSTO Defence Science and Technology Organisation, Edinburgh, Australia, July, 2011.

[62] Trust-hub, [Online]. Available: https://www.trust-hub.org/.

[63] D. Stefan, C. Mitchell and C. G. Almenar, "CSAW Embedded System Challenge 2008," [Online]. Available: isis.poly.edu/~vikram/cooper.pdf.

[64] A. Baumgarten, M. Steffen, M. Clausman and J. Zambreno, "A case study in hardware Trojan design and implementation," *International Journal of Information Security*, vol. 10, no. 1, pp. 1-14, 2011 .

[65] L. Fiorin, G. Palermo, S. Lukovic and C. Silvano, "A data protection unit for NoC-based architectures," in *Proc. of 5th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Sept. 2007.

[66] W. Chouchene, B. Attia, A. Zitouni, N. Abid and R. Tourki, "A low power network interface for network on chip," in *Proc. of 8th International Multi-Conference on Systems, Signals and Devices (SSD)*, March 2011.

[67] J.-H. P. S.-C. Lee, "A Practical Design and Implementation of On-Chip NI for Integrating Bus Based IP Legacies," in *Proc. of 6th WSEAS International Conference on Instrumentation, Measurement, Circuits & Systems*, Hangzhou, China, April 2007.

[68] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test*, vol. 27, no. 1, pp. 10-25, 2009.

[69] B. Zitouni and R. Tourki, "Design and implementation of network interface compatible OCP For packet based NOC," in *Proc. of 5th International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, March 2010.

[70] J. Santos and Y. Fei, "Designing and implementing a Malicious 8051 processor," in *Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2012.

[71] X. Wang, M. Tehranipoor and J. Plusquellic, "Detecting malicious inclusions in secure

hardware: Challenges and solutions," in *Proc. of IEEE International Workshop on Hardware-Oriented Security and Trust*, June 2008.

[72] A. Das, G. Memik, J. Zambreno and A. Choudhary, "Detecting/preventing information leakage on the memory bus due to malicious hardware," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2010.

[73] M. Dehyadgari, M. Nickray, A. Afzali-kusha and Z. Navabi, "Evaluation of Pseudo Adaptive XY Routing Using an Object OrientedModel for NOC," in *Proc. of 17th International Conference on Microelectronics*, March 2005.

[74] S. Narasimhan, R. Chakraborty and S. Bhunia, "Hardware IP Protection During Evaluation Using Embedded Sequential Trojan," *IEEE Design & Test*, vol. 29, no. 3, pp. 70-79, 2012.

[75] S. Bhasin, J.-L. Danger, S. Guilley, X. Ngo and L. Sauvage, "Hardware Trojan Horses in Cryptographic IP Cores," in *Proc. of 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Aug. 2013.

[76] W. Shi, J. Fryman, G. Gu, H.-H. Lee, Y. Zhang and J. Yang, "InfoShield: a security architecture for protecting information usage in memory," in *Proc. of International Symposium on High-Performance Computer Architecture*, Feb. 2006.

[77] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. of Design Automation Conference*, 2001.

[78] J. Rajendran, E. Gavas, J. Jimenez, V. Padman and R. Karri, "Towards a comprehensive and systematic classification of hardware Trojans," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2010.

[79] F. Wolff, C. Papachristou, S. Bhunia and R. Chakraborty, "Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme," in *Proc. of Design, Automation and Test in Europe (DATE)*, March 2008.

[80] D. Bertozzi and B. L., "Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip," *IEEE Circuits and Systems*, vol. 4, no. 2, pp. 18-31, 2004.